INTEGRATED LOCATION, ROUTING AND SCHEDULING PROBLEMS: MODELS AND ALGORITHMS

by

Zeliha Akça

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial and Systems Engineering

Lehigh University

January 2010

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

_____
Date

_____
Dr. Theodore K. Ralphs
Dissertation Director

_____
Accepted Date

Committee:

_____
Dr. Theodore K. Ralphs, Chairman

_____
Dr. Rosemary T. Berger

_____
Dr. Jeffrey T. Linderoth

_____
Dr. Karen Smilowitz

Acknowledgments

I would like to thank a number of people who have supported me in any way during my Ph.D. study. First of all, I am grateful to my parents and my brother for their unconditional love and support in my life. I would like to thank to other members of my family for their support to my parents while I have been abroad for the PhD study.

I would like to thank to my previous dissertation advisor Dr. Rosemary Berger for helping me to start and develop my research and for her support and directions during my graduate study. I would like to thank to my dissertation advisor, Assoc. Dr. Ted Ralphs, for accepting being my advisor, helping me patiently to construct and finalize my Ph.D. research, assisting me to provide better outputs, challenging me with different views and for his continuous support. I have learned a great deal from them, I feel very much indebted. I would also like to thank to my other members of my dissertation committee for their evaluations, directions and insightful comments.

I am grateful to the P.C. Rossin School of Engineering and the Department of Industrial and Systems Engineering at Lehigh University for providing me financial support during my education. Many thanks also go to those who arranged the financial support. Special thanks to Dr. Larry Snyder for hiring me as a research assistant in his project and helping me to get industrial experience. I would like to thank to the faculty in Industrial and Systems Engineering Department for their invaluable assistance in my training. I should thank to Dr. Jeffrey Linderoth for being very supportive throughout my graduate study. I am truly happy to know him. I also want to thank him for his help to solve problems we had with MINTO.

I would like to thank to my friends, Mehmet Oguz Atan, Tolga Seyhan, Umit Ozkan, Wasu Glankwamdee, Kamonkan Laksana and many others who I could not list all here for their friendship and unconditional assistance when needed. I would like to thank to my friends Tuba Alim Kilic, Selin Bilgin Ozpeynirci, Ozgul Baysar, Nilgun Esin Aksoy, and Zehra Bal who have supported me despite the distance between us. I would like to thank to Ashutosh Mahajan for his friendship, professional collaboration and his help in solving software and hardware problems I had in my studies. Special thanks to my fellow friends Banu Gemici Ozkan, Menal Guzelsoy, Pinar Guvelioglu, Zumbul Bulut, Ruken Duzgun and Berrin Aytac for being with me at Lehigh and sharing my key moments. Their friendship means a lot to me. I also want to thank them for their professional collaboration during my education. I am grateful to Menal Guzelsoy for being available for 7 days and 24 hours in order to give me technical support in hardware and software related problems.

Finally, I would like to thank to Rita Frey and Kathy Rambo, coordinators in Industrial and Systems Engineering Department, for their assistance since I got acceptance to the PhD program.

Contents

List of Tables

List of Figures

Abstract

In this research, we investigate location, routing and scheduling problems, a class of problems in which the decisions of facility location, vehicle routing, and route assignment are optimized simultaneously. The problem considers the interdependency between facility location and vehicle routing decisions and also allows the assignment of multiple routes to a single vehicle, resulting in a solution that requires fewer vehicles. We refer to this integrated problem, which is in the complexity class NP-hard, as the location routing and scheduling problem (LRSP).

For a version of this problem with capacitated facilities and time- and capacity-limited vehicles, we present two formulations, one graph-based and one set partitioning-based. We derive the theoretical relationship between these two formulations by applying Dantzig-Wolfe decomposition to the graph-based formulation. We show the equivalence of the set partitioning-based model and the reformulation of the graph-based formulation obtained by Dantzig-Wolfe decomposition. In addition, we computationally compare the bounds yielded by the LP relaxations of the different formulations of the LRSP and demonstrate the strength of the LP relaxation of the set partitioning-based model and some simple valid inequalities introduced for this model.

By adopting a column generation approach derived from the set partitioning-based formulation of the problem, we provide exact and heuristic solution algorithms for the problem. We describe two versions of a branch-and-price algorithm, a standard one-stage version and a two-stage version. The two-stage version is an extended variant of the one-stage algorithm based on the idea of ”restart.” It has an initialization phase that provides a good upper bound and a pool of effective

columns. We evaluate and computationally demonstrate the performance of the algorithms we design using instances up to 40 customers and various parameter settings. We extend our computational study in order to assess the benefit of integrating optimization of facility location, vehicle routing and vehicle scheduling decisions by comparing the LRSP with some variations of the sequential optimization of these decisions. We obtain results confirming our interest in solution of the LRSP.

We extend the branch-and-price algorithm to include methods for dynamically generating valid inequalities. We investigate classes of inequalities valid for polytopes related to the LRSP polytope; discuss their validity for the problem and their integration into the branch-and-price algorithm. In addition, we explore possible ways of adapting the approach of generating valid inequalities using the disjunctive procedure. We provide a methodology to derive disjunctive inequalities in a branch-and-price algorithm and describe some implementations of the methodology for our algorithm. We computationally compare these valid inequalities and discuss their effect to the performance of the branch-and-price algorithm.

Finally, we discuss how we can modify the presented solution approaches for a location and routing problem (LRP), which is a special case of the LRSP. We present computational results for instances up to 40 customers and for instances of certain special cases previously considered in the literature to demonstrate the effectiveness of the solution approach empirically.

Chapter 1

Introduction

Logistics consists of the planning and controlling of the transportation and storage activities in a system that produces and distributes goods and/or services. It is one of the most important functions of many companies. The design of a distribution system begins with the questions of where to locate the facilities and how to allocate customers to the selected facilities. Furthermore, determining how to distribute goods to customers from these facilities and how to use vehicles and/or distribution resources efficiently are important decisions that arise in the design of logistics systems.

In this research, we focus on a class of problems that integrates the decisions of facility location, vehicle routing, and route assignment and seeks to minimize the total cost. We refer to such problems as location, routing and scheduling problems (LRSPs). The LRSP generalizes and subsumes several well-studied problem classes, such as the location and routing problem (LRP), the vehicle routing problem (VRP), the multi-depot VRP (MDVRP), and the VRP with multiple trips (VRPMT).

Given a set of candidate facility locations and a set of customer locations, the objective of the LRSP is to select a subset of the facilities, construct a set of delivery routes, and assign routes to vehicles in such a way as to minimize total cost, subject to satisfaction of the system's constraints. The total cost may include both fixed costs and operating costs of facilities and vehicles.

The LRSP arises in the context of many delivery problems, such as gas, oil, food, beverage, bill and newspaper delivery problems. In practical applications of the LRSP, various constraints need to be considered. For example, facilities and vehicles may have capacities, the length of routes may be restricted with time limits, service of customers may include both deliveries and pick-ups, service to customers might be restricted to occur within time windows, different types of vehicles might be used, customers might have priorities, and some data such as customer demands or travel times might be uncertain. In this research, we consider an LRSP with capacity constraints on the facilities and on the vehicles, as well as time constraints on the vehicles. We assume that all data is deterministic, and the problem is either purely a pick-up problem or purely a delivery problem.

1.1 Motivation

The motivation behind the integration of the facility location, vehicle routing and vehicle scheduling decisions is to improve the efficiency of the system and yield more accurate and cost-effective solutions. In the literature, it was realized early that the location decision and the routing decision are interdependent when customers demand less-than-truckload quantities (Webb (1968) and Salhi and Rand (1989)). Therefore, if it is possible to serve customers through routes making multiple stops, the assumption of single-customer routes in location models will not accurately capture the transportation cost. Gas and oil delivery systems, distribution of goods to retailers, and newspaper and bill delivery systems are examples of such systems that require routes with multiple stops.

Location and routing models seek to minimize total cost by simultaneously selecting a set of facilities and constructing a set of delivery routes that satisfy the specified system constraints. LRPs, however, implicitly assume that each vehicle covers exactly one route. This may potentially overestimate the number of vehicles required and the associated distribution cost. In many cases, it is possible to serve multiple routes with a single vehicle, in which case the decision of assigning routes to vehicles becomes interdependent with the location and routing decisions. Eilon (1977)

reports that fixed costs are equal to the eighty percent of total vehicle costs. Therefore, models integrating the scheduling decision and considering the fixed costs of the vehicles may yield more cost efficient solutions. In Section 5.6, we empirically demonstrate the value of integrating scheduling decision with the location and routing decisions by comparing the LRSP solution with the LRP solution and the solution to the optimization of scheduling decision based on the resulting LRP solution. Furthermore, some systems require different types of vehicles, which may have different costs than the multi-purpose vehicles. In such cases, the number of customized vehicles is an important decision for the company. Similarly, if the service to the customer requires a trained worker who is more expensive than a regular driver, then the company will want to consider training costs, which are also fixed costs.

Why is capturing the costs more accurately important? Major components of logistics cost are warehousing costs including inventory carrying costs and transportation costs. Based on estimates for the U.S. in 2002, transportation costs were $577 billion, and inventory carrying costs and warehousing costs were $298 billion. The total logistics costs were $910 billion, which was equivalent to 8.7% of the U.S. gross domestic product in 2002 (MacroSy Research and Technology (2005)). Undeniably, logistics costs are very important for the U.S. economy as well as for a company. Since the contribution of logistics activities to the GDP is large, improvements in the efficiency of the logistics activities will contribute to the growth of the U.S. GDP. Even a small amount of improvement in distribution systems can result in a substantial amount of savings.

Another motivation to consider scheduling decision together with the location and routing decisions is to be able include more system constraints to the model. In distribution systems in which the company has a constant fleet size, the number of vehicles should be considered in the model. To be able to cover all customers, it might be necessary to use a vehicle for more than one route. Therefore, the decision of scheduling of vehicles must be considered with the decisions of location and routing. In some systems, there are time constraints for the customers or the vehicles. Delivery of some items such as perishable foods, newspapers and bills are time sensitive. In addition, the overtime for the workers might be expensive, hence the company is not willing to

pay overtime driver hours. Therefore, while designing the model, time limit for the vehicle routes must be considered.

1.2 Background

In this section, we provide some definitions, briefly explain some terminology and the associated notation that is used in this thesis. The following background information was prepared based on several sources, mostly from Nemhauser and Wolsey (1989), Linderoth (2005), Thomas (2005) and Perregaard and Balas (2001).

1.2.1 Notation

We denote the sets of all (nonnegative) real numbers, integers and rational numbers with symbols $\mathbb{R}$ ($\mathbb{R}_+$), $\mathbb{Z}$ ($\mathbb{Z}_+$) and $\mathbb{Q}$ ($\mathbb{Q}_+$), respectively. The set of all (nonnegative) real vectors with dimension n for some positive scalar n is represented by $\mathbb{R}^n$ ($\mathbb{R}^n_+$). Similar notation is used to define sets of all integer and rational vectors.

We use uppercase mathematical (italic) letters such as A, B, I, J to denote matrices and sets. When we introduce such notation, we indicate what it represents. We use lowercase mathematical (italic) letters such as d, n, s, x, for scalars and vectors. The notation $c^\top$ is used to represent the transpose of vector c, and the notations $\lfloor d \rfloor$ and $\lceil d \rceil$ are used to represent the floor and the ceiling of the scalar d. Uppercase calligraphic characters such as $\mathcal{P}$, $\mathcal{K}$, are used to denote polyhedra and polyhedral sets. The notation $conv(C)$ represents the convex combination of all points in set C (i.e., the convex hull of C). We use notation (α, β) where $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$ to represent an inequality in the form of $\alpha x \geq \beta$ for some vector $x \in \mathbb{R}^n$.

1.2.2 Polyhedra

A set $R \subseteq \mathbb{R}^n$ is called *convex* if for every $x, y \in R$ and for all $\lambda \in [0, 1]$, $\lambda x + (1 - \lambda)y \in R$. Let $x^1, x^2, ..., x^k \in \mathbb{R}^n$. The vector $\sum_{i=0}^{k} \lambda_i x^i$ is called a *convex combination* of vectors $x^1, x^2, ..., x^k$

for some $\lambda \in \mathbb{R}^k_+$ satisfying $\sum_{i=0}^k \lambda_i = 1$. The set of all convex combinations of members of set $R \subset \mathbb{R}^n$ is called *convex hull of* R, i.e., $conv(R)$.

A set $\mathcal{P} \subseteq \mathbb{R}^n$ of the form $\{x \in \mathbb{R}^n | Ax \geq d\}$ for a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $d \in \mathbb{R}^m$ is called a *polyhedron*. All polyhedra are convex sets. A polyhedron $\mathcal{P}$ is said to be *bounded* if $|x| \leq k$ is true for some constant k and for all $x \in \mathcal{P}$. A bounded polyhedron is called a *polytope*.

A *valid inequality* for $\mathcal{P} = \{x \in \mathbb{R}^n | Ax \geq d\}$ is a pair (α, β), where $\alpha \in \mathbb{R}^n$ is the coefficient vector and $\beta \in \mathbb{R}$ is the right-hand side such that $\alpha x \geq \beta$ for all $x \in \mathcal{P}$. If (α, β) is a valid inequality for $\mathcal{P}$, $\mathcal{F} = \{x \in \mathcal{P} | \alpha x = \beta\}$ is called a *face* of $\mathcal{P}$. If $\mathcal{F}$ is nonempty and $\mathcal{F} \neq \mathcal{P}$, then $\mathcal{F}$ is called a *proper* face of $\mathcal{P}$. Maximal proper faces are called *facets*. A proper face is a facet if its dimension is one less than the dimension of the polyhedron. The *dimension* of a polyhedron is equal to the maximum number of affinely independent points contained in the polyhedron. The points $x^1, x^2, ..., x^k \in \mathbb{R}^n$ are called *affinely independent* if $\lambda = 0$ is the unique solution to $\sum_{i=0}^k \lambda_i x^i = 0$ and $\sum_{i=0}^k \lambda_i = 0$ for $\lambda \in \mathbb{R}^k$.

1.2.3 Linear Programming

A *linear program* (LP) is the problem of minimizing (or maximizing) a linear objective function subject to a set of linear constraints. To be more precise, the general form of an LP is $z_L = \min\{c^\top x | Ax \geq d\}$, where $A \in \mathbb{Q}^{m \times n}$ is the *constraint matrix*, $c \in \mathbb{Q}^n$ is the *objective function* vector and $d \in \mathbb{Q}^m$ is the *right hand side* vector. The vector $x \in \mathbb{R}^n$ is called the vector of *decision variables*. Any vector $\hat{x} \in \mathcal{P}_\mathcal{L}$ is called a *feasible solution* to the LP, where $\mathcal{P}_\mathcal{L}$ is the underlying polyhedron $\mathcal{P}_\mathcal{L} = \{x \in \mathbb{R}^n | Ax \geq d\}$ (also called the *feasible region*).

The LP is said to be *infeasible* if $\mathcal{P}_\mathcal{L}$ is empty. In that case, we set $z_L = \infty$. The LP is said to be *unbounded* if $z_L = -\infty$. If it is not infeasible or unbounded, the LP has a finite optimal. z_L is called the *optimal objective (function) value* and $x_L \in \mathcal{P}_\mathcal{L}$ is called an *optimal solution* if $z_L = c^\top x_L \leq c^\top x$ for all $x \in \mathcal{P}_\mathcal{L}$.

The LP $z_D = \max\{y^\top d | y^\top A = c, y \geq 0\}$ is called the *dual* of $\min\{c^\top x | Ax \geq d\}$ and $y \in \mathbb{R}^m$ is called the *dual vector*. Let $\mathcal{P}_D = \{y \in \mathbb{R}^m_+ | y^\top A = c\}$. For $\hat{y} \in \mathcal{P}_D$, vector $c - \hat{y}A$

is called the *reduced cost vector* corresponding to variable vector $x \in \mathcal{P}_L$ and $\hat{y}^\top d \leq c^\top x$ for all $x \in \mathcal{P}_L$. This property of LPs is referred to as the *weak duality property* in the literature. Furthermore, if an LP has a finite optimal value, then so does the associated dual LP and these optimal values are equal (referred to as *strong duality property*). If $z_L = -\infty$, then $z_D = -\infty$ (infeasible). If $z_D = \infty$ (unbounded), then $z_L = \infty$. Both the primal and dual LPs can also be infeasible.

1.2.4 Mixed Integer Linear Programming

A *mixed integer linear program* (MILP) is the problem of minimizing or maximizing a linear objective function subject to a set of linear constraints and integrality restrictions on some or all of the variables. More specifically, an MILP model (which we refer to as (P) in the rest of the thesis) can be formulated as follows

$$\begin{aligned} z_{IP} = \text{Minimize} \quad & c^\top x \\ \textbf{(P)} \qquad \text{subject to} \quad & Ax \geq d, \\ & x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+, \end{aligned}$$

where $A \in \mathbb{Q}^{m \times n}$ is the constraint matrix, $c \in \mathbb{Q}^n$, $d \in \mathbb{Q}^m$ are rational vectors and $x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+$ is a vector of decision variables. Without loss of generality, variables that are indexed from 1 to r are called *integer variables*, and those indexed from $n-r+1$ to n are called *continuous variables*. If $\mathcal{P} = \{x \in \mathbb{R}^n_+ | Ax \geq d\}$ is the polyhedron defined by the linear constraints of (P), the set $S = \{x \in \mathbb{Z}^r \times \mathbb{R}^{n-r} | x \in \mathcal{P}\}$ is called the *feasible solution set* of (P) and $x \in S$ is called a feasible solution to (P).

If $S = \emptyset$, then (P) is said to be infeasible, and we set $z_{IP} = \infty$. Otherwise (P) is said to be feasible. If $z_{IP} = -\infty$, (P) is said to be unbounded. If (P) is feasible and z_{IP} is finite, than (P) has a finite optimal. z_{IP} is the optimal objective value and $x_{IP} \in S$ is an optimal solution if $z_{IP} = c^\top x_{IP} \leq c^\top x$ for all $x \in S$.

If $r = n$, then (P) is called a *(pure) integer linear program* (ILP). The LP that is obtained by relaxing all integrality restrictions on the variables in (P) is called the *LP relaxation* of (P). We refer to the value of the LP relaxation of (P) as z_{LP} (i.e., $z_{LP} = \min_{x \in \mathcal{P}} c^\top x$) and the solution vector as $x_{LP} \in \mathcal{P}$.

1.2.5 Cutting-Plane Procedure

The cutting-plane procedure is a methodology used to obtain a lower bound for the optimal solution value of an MILP by generating approximations of the feasible solution set associated with the MILP. Consider the MILP (P). The algorithm iteratively provides approximations of the feasible solution set S as the intersection of half-spaces associated with inequalities valid for S. A *cutting plane* is a valid inequality that separates a given solution, say $\bar{x} \in \mathcal{P}$, such that $\bar{x} \notin S$ from S. In other words, if (α, β) is such a cutting plane, then $\alpha x \geq \beta$ for all $x \in S$ and $\alpha \bar{x} < \beta$. The problem of determining whether $\bar{x} \in S$ or finding a valid inequality (α, β) such that $\alpha \bar{x} < \beta$ is called the *separation problem*. In the standard procedure, $\mathcal{P}$ is the initial approximation of S and z_{LP} is the corresponding lower bound for z_{IP}. The algorithm starts by generating a valid inequality (α, β) that is violated by x_{LP}, the solution to the LP $\min_{x \in \mathcal{P}} c^\top x$. The polyhedron obtained by intersecting the half-space associated with (α, β) and $\mathcal{P}$ is a better approximation of S and may provide a better bound for z_{IP}. This process continues iteratively until no violated valid inequality is obtained or a solution of S is found.

In order to present an outline of the algorithm, we define some notation. Let z_{LB} represent the lower bound on the optimal objective function value of the MILP that is being handled in the algorithm. $\mathcal{P}^i$ denotes the underlying polyhedron at iteration i of the cutting-plane algorithm; P^i denotes the MILP defined with feasible region $\mathcal{P}^i$ and the integrality restrictions for the variables; x^i_{LP} and z^i_{LP} denote the solution and the objective function value of the LP relaxation of P^i, respectively; and (α^i, β^i) is the valid inequality obtained by solving the separation problem for x^i_{LP}. An outline of a cutting-plane algorithm is presented in Figure 1.1. Next, we briefly describe some commonly used classes of valid inequalities that are employed in our study.

CUTTING-PLANE
Input: An instance (P).
Output: A lower bound z_{LB} for z_{IP}.

0. Set $i \leftarrow 0$ and $z_{LB} \leftarrow -\infty$. Let $\mathcal{P}^0 \leftarrow \mathcal{P}$.
1. Solve the LP relaxation of P^i with feasible region $\mathcal{P}^i$. Set $z_{LB} \leftarrow z^i_{LP}$.
   - If $z^i_{LP} = \infty$, then P^i is infeasible, STOP.
   - If $z^i_{LP} = -\infty$, then P^i is unbounded, STOP.
   - If $x^i_{LP} \in S$, then x^i_{LP} is the optimal solution and z^i_{LP} is the optimal solution value. STOP.
   - Otherwise, go to next step.
2. Solve the separation problem for x^i_{LP}.
   - If a violated inequality (α^i, β^i) is obtained, set $\mathcal{P}^{i+1} \leftarrow \mathcal{P}^i \cap \{x \in \mathbb{R}^n | \alpha^i x \geq \beta^i\}$ and $i \leftarrow i + 1$. Return to step 1.
   - Otherwise, STOP.

Figure 1.1: Outline of the Cutting Plane Algorithm

Chvátal-Gomory (CG) Inequalities. *Chvátal-Gomory (CG) valid inequalities* (described by Gomory (1963) and Chvátal (1973)) are a class of valid inequalities that are obtained by taking linear combinations of the inequalities in the current model and applying a rounding procedure. Using the integrality of the variables, the rounding procedure leads to integer right and left hand sides in any inequality. Consider the MILP (P) with $r = n$. A CG inequality is a valid inequality for S which is of the form $\lceil \lambda^\top A \rceil x \geq \lceil \lambda^\top d \rceil$, where $\lambda \in \mathbb{R}^n_+$. The *CG rank* of any original inequality and any inequality dominated by a nonnegative linear combination of the original inequalities, e.g., $\lambda^\top Ax \geq \lambda^\top d$ for $\lambda \in \mathbb{R}^n_+$, is said to be 0. An inequality that can be generated as a CG inequality based on rank 0 inequalities but is not rank 0 itself has CG rank 1. Inequalities with higher rank can be obtained recursively from the lower rank inequalities.

Disjunctive Valid Inequalities. A disjunction is a logical operator that evaluates TRUE whenever one or more of its operands evaluates TRUE. In the context of MILP, the disjunctions most

commonly arising are those whose operands are systems of linear equalities or inequalities. More specifically, such a disjunction is defined by s systems over set S:

$$B^1x \geq b^1 \ \vee \ B^2x \geq b^2 \quad \vee \quad ... \quad \vee \quad B^sx \geq b^s \quad \forall x \in S, \tag{1.1}$$

where $B^k \in \mathbb{Q}^{m^k \times n}$ is a constraint matrix, $b^k \in \mathbb{Q}^{m^k}$ is a right hand side vector for some integer $m^k > 0, \forall k = 1, .., s$. The associated disjunction described by (1.1) results in true if and only if $B^k\hat{x} \geq b^k$ is true for some $s \geq k \geq 1$ and $\hat{x} \in S$.

In many solution approaches for general MILPs such as branch-and-cut, the concept of valid disjunction is fundamental. Consider the MILP (P). The disjunction described by (1.1) is called *valid* for (P), if

$$S \subseteq \cup_{k=1}^{s}\{x \in \mathbb{R}^n_+ | Ax \geq d, B^kx \geq b^k\}.$$

The generation of cutting planes based on disjunctions and the idea of disjunctive programming were introduced in Balas (1974) and Balas (1979). Generally speaking, disjunctive programming is finding the optimal value of an objective function over a finite union of polyhedra constructed based on a combination of conjunctive statements (connected with logical operator "AND") and disjunctive statements. Here, we focus on disjunctive statements. We explain the methodology based on (P) using the valid disjunction for (P) of the form (1.1).

Using the s systems of inequalities in the valid disjunction, a relaxation of (P) (denoted by (P_{DR})) can be defined in the form of a disjunctive program:

$$\begin{aligned} & \text{Minimize} && c^\top x \\ \textbf{(P}_{\textbf{DR}}\textbf{)} \qquad & \text{subject to} && Ax \ \geq \ d, \\ & && \vee_{k=1}^{s} B^k x \ \geq \ b^k, \\ & && x \ \geq \ 0. \end{aligned}$$

We interpret the above program as follows. With term k of the disjunction, we associate a polyhedron.

$$\mathcal{P}^k = \{x \in \mathbb{R}^n_+ | Ax \geq d, B^k x \geq b^k\} \ \forall k = 1..s.$$

(P_{DR}) can then be interpreted as an optimization problem over the union of s polyhedra.

$$\begin{aligned} z_{DR} = & \text{Minimize} \quad c^\top x \\ & \text{subject to} \quad x \in \mathcal{P}_{DR}, \end{aligned}$$

where $\mathcal{P}_{DR} = \cup_{k=1}^{s} \mathcal{P}^k$. Since ($\text{P}_{\text{DR}}$) is a relaxation of (P), any inequality valid for $\mathcal{P}_{DR}$ is also valid for S. The aim of the disjunctive cut generation methodology is to derive an inequality, (α, β) that is valid for $\mathcal{P}_{DR}$ and is violated by some feasible solutions of the LP relaxation of (P).

Balas (1979) showed that (α, β) is valid for $\mathcal{P}_{DR}$ if and only if there exist multipliers $u^k \in \mathbb{Q}^m$ and $v^k \in \mathbb{Q}^{m^k}$ such that

$$\begin{aligned} u^k A \ + \ v^k B^k \quad &\leq \quad \alpha \quad \forall k = 1, .., s, \\ u^k b \ + \ v^k b^k \quad &= \quad \beta \quad \forall k = 1, .., s, \\ u^k, v^k \quad &\geq \quad 0 \quad \forall k = 1, .., s. \end{aligned}$$

Hence, to generate an inequality (α, β) that is maximally violated by x_{LP} and valid for $\mathcal{P}_{DR}$, the

following optimization model, called the *cut generating LP* (CGLP) is solved.

$$\begin{aligned}
& & \text{Minimize} \quad & \alpha^\top x_{LP} - \beta & & & & & (1.2)\\
\textbf{(CGLP)} & & \text{subject to} \quad & u^k A + v^k B^k & \leq & \alpha & \forall k = 1, .., s, & (1.3)\\
& & & u^k b + v^k b^k & = & \beta & \forall k = 1, .., s, & (1.4)\\
& & & \sum_{k=1}^{s}\sum_{i=1}^{m} u_i^k + \sum_{k=1}^{s}\sum_{i=1}^{m^k} v_i^k & = & 1, & & (1.5)\\
& & & u^k, v^k & \geq & 0 & \forall k = 1, .., s. & (1.6)
\end{aligned}$$

The objective function (1.2) seeks to maximize the violation for x_{LP}. If the optimal objective function value is negative, then x_{LP} violates the inequality (α^*, β^*) which is the optimal solution. Constraints (1.3) and (1.4) ensure that any feasible solution (α, β) of this problem is a valid inequality for $\mathcal{P}_{DR}$. Constraint (1.5) is added to normalize the cut to be generated. The purpose of the normalization constraint is to truncate the cone defined by constraints (1.3) and (1.4) to a polytope. The normalization removes the unboundedness of the model and transforms extreme rays to extreme points. Here, we use one of the common normalization constraints, different versions of normalization constraints are described in the literature. Some examples are $\alpha-$normalization (i.e.,$\sum_{i=1,..,n} |\alpha_i| = 1$) and $\beta-$normalization (i.e.,$|\beta| = 1$) (Balas et al. (1996)).

As noted in Caprara and Letchford (2003) (based on the results in Balas et al. (1993) and Balas et al. (1996))), for a fixed disjunction, separation of disjunctive cuts can be accomplished in polynomial time by solving the above (CGLP). In general, it is common to use disjunctions including two terms consisting of linear inequalities. One of the most common and successful procedures based on this principle is referred to as *lift-and-project* (see Balas et al. (1993) and Balas et al. (1996)). In lift-and-project, valid inequalities are derived from disjunctions involving bounds on single binary variables of the form $(x_i \leq 0) \vee (x_i \geq 1)$ for some index i of a binary variable. Lift-and-project cuts are one of the widely-used general-purpose cutting planes for 0-1 mixed integer programming models.

If the disjunction used to derive a given cut is of the more general form $(\pi x \leq \pi_0) \vee (\pi x \geq \pi_0 + 1)$ for some integer vector $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ such that $\pi_i = 0$ for $i = r+1, .., n$ (continuous variables), the cut is called a *split cut* (Cook et al. (1990) and Balas and Saxena (2008)). Caprara and Letchford (2003) proved that the separation problem for split cuts is strongly NP-Complete. Many general-purpose classes of valid inequalities for MILPs are in the broad class of split cuts. The well-known valid inequalities Gomory (Gomory (1963)) and the mixed integer rounding inequalities (Nemhauser and Wolsey (1990)), for example, have been shown to be in the class of split cuts (Cornuejols and Li (2001) and Cornuejols (2006)).

In addition to these studies that derive valid inequalities from valid disjunctions, Letchford (2001) showed that many facet-inducing inequalities for well-known combinatorial optimization problems are actually disjunctive valid inequalities derived from certain simple disjunctions. They used these results to prove that the separation problems for these facet-inducing inequalities are solvable in polynomial time. They provided results for some facets of the clique partitioning problem, the maximum cut problem, the acyclic sub digraph problem, the linear ordering problem, the asymmetric traveling salesman problem and the set covering problem.

1.2.6 Branch-and-Bound Algorithms

The most widely used class of algorithms for solving MILPs is the *branch-and-bound algorithms*, which employ a divide-and-conquer approach. The algorithm divides the set of feasible solutions into subsets to implicitly enumerate all feasible solutions of the problem. Optimizing the objective function for each subset results in *subproblems* of the original problem. Consider the MILP (P). The basic idea is to partition S into k subsets S_1, ..., S_k and recursively optimize over each subset:

$$\min_{x \in S} c^\top x \; = \; \min_{i=1..k} \{\min_{x \in S_i} c^\top x\}.$$

Partitioning a feasible set into smaller subsets is called *branching*. To avoid the complete enumeration of all feasible solutions, the algorithm iteratively calculates *upper and lower bounds* on the

optimal objective function value for each subproblem. Any feasible solution $\hat{x} \in S$ to the problem provides an upper bound. The most common lower-bounding procedure used in the algorithm is to solve an LP relaxation of the subproblem. In such an LP-based branch-and-bound algorithm, the LP relaxation of the original problem is first solved to obtain x_{LP}. If $x_{LP} \in S$, then it is also optimal for (P) and the algorithm terminates. If the LP relaxation is unbounded or infeasible, then (P) is also unbounded or infeasible. Otherwise, we need to branch and solve the resulting subproblems, recursively.

The development of the algorithm can be visualized by constructing an associated tree (which is referred to as branch-and-bound tree) in which each node represents a subproblem. The root node represents the MILP that is going to be solved and successors represent subproblems obtained by branching. In order to describe the algorithm and present an outline, we define some notation. We will use the same notation in the rest of the thesis. P^i denotes the subproblem associated with node i and defined over subset S^i obtained by applying terms of the branching disjunctions associated with node i to set S. $\mathcal{P}^i$ denotes the polyhedron defined by the linear constraints of the subproblem P^i. x^i_{LP} and z^i_{LP} denote the LP relaxation solution and objective function value for subproblem P^i, respectively. For consistency, we let P^0 denote the subproblem at the root node, which is (P) and the set S^0 denotes the feasible solution set S. We use z_{UB} and z_{LB} to represent the global upper and lower bounds on the optimal objective function value for (P). L represents the candidate list that includes the subproblems to be processed. The subproblems generated by partitioning the feasible solution set of another subproblem (say subproblem P^i) are referred to as the *children* of P^i, while P^i is referred to as the *parent*. An outline of a branch-and-bound algorithm is presented in Figure 1.2.

1.2.7 Branch-and-Cut Algorithms

Another solution methodology applied to MILP is *branch and cut*, which is also a divide-and-conquer approach and is a combination of the branch-and-bound algorithm and the cutting-plane procedure. The basic idea of the algorithm is the same as the branch-and-bound algorithm. The

BRANCH-BOUND
Input: An instance (P).
Output: z_{IP} and x_{IP}.

0. Set $z_{UB} \leftarrow \infty$ or to the value of a known feasible solution, $z_{LB} \leftarrow -\infty$, and $L \leftarrow \{\mathrm{P}^0\}$.
1. Select a subproblem P^i from L. Set $L \leftarrow L \backslash \{\mathrm{P}^i\}$.
2. Solve the LP relaxation of P^i with feasible region $\mathcal{P}^i$ and obtain z^i_{LP}.
   - If $z^i_{LP} \geq z_{UB}$, go to step 5.
   - If $x^i_{LP} \in S$, then set $z_{UB} \leftarrow \min\{z^i_{LP}, z_{UB}\}$ and go to step 5.
   - Otherwise, go to next step.
3. Branch (partition S^i into subsets). Define subproblems associated with subsets. Add new subproblems to L.
4. $z_{LB} = \min\{z^k_{LP} | \mathrm{P}^k$ is a parent problem for $\mathrm{P}^i \in L\}$.
5. If $z_{LB} \geq z_{UB}$ or $L = \emptyset$, then set $z_{IP} \leftarrow z_{UB}$ (if there is a solution vector that yields z_{UB}, then it is x_{IP}), and STOP. Otherwise, return to step 1.

Figure 1.2: Outline of the Branch-and-Bound Algorithm

feasible solution set is partitioned into subsets and each resulting subproblem is optimized, recursively. Recall that in an LP-based branch-and-bound algorithm, a lower bound is determined by solving the LP relaxation of each subproblem. In a branch-and-cut algorithm, this lower bound is strengthen by dynamically generating valid inequalities for each subproblem.

One major challenge in a branch-and-cut procedure is the difficulty of the separation problems to be solved to generate the valid inequalities. In most cases, heuristic approaches are employed in the solution of the separation problem. Another challenge is to determine a class of valid inequalities that would improve the LP relaxation bound most. Effectiveness of the algorithm depends on the classes of the valid inequalities to be generated and the efficiency of the separation routines required for those.

The steps of a branch-and-cut algorithm are similar to BRANCH-BOUND except for one additional step in which valid inequalities are generated. We use the same notation used for BRANCH-BOUND and define some additional notation. We denote the set of valid inequalities generated

BRANCH-CUT
Input: An instance (P).
Output: z_{IP} and x_{IP}.

0. Set $z_{UB} \leftarrow \infty$ or to the value of a known feasible solution, $z_{LB} \leftarrow -\infty$, and $L \leftarrow \{\mathrm{P}^0\}$.
1. Select a subproblem P^i from L. Set $L \leftarrow L \backslash \{\mathrm{P}^i\}$.
2. Solve the LP relaxation of P^i with feasible region $\mathcal{P}^i$ and obtain z^i_{LP}.
   - If $z^i_{LP} \geq z_{UB}$, go to step 6.
   - If $x^i_{LP} \in S$, then set $z_{UB} \leftarrow \min\{z^i_{LP}, z_{UB}\}$ and go to step 6.
   - Otherwise, go to next step.
3. If desired, search for valid inequalities violated by x^i_{LP}. Let F^i be the set of inequalities generated.
   - If $F^i \neq \emptyset$, set $\mathcal{P}^i \leftarrow \mathcal{P}^i \cap \{x \in \mathbb{R}^n | D^i x \geq d^i\}$. Return to step 2.
   - Otherwise, proceed.
4. Branch (partition S^i into subsets). Define subproblems associated with subsets. Add new subproblems to L.
5. $z_{LB} = \min\{z^k_{LP} | \mathrm{P}^k$ is a parent node for problem $\mathrm{P}^i \in L\}$.
6. If $z_{LB} \geq z_{UB}$ or $L = \emptyset$, then set $z_{IP} \leftarrow z_{UB}$ (if there is a solution vector that yields z_{UB}, then it is x_{IP}), and STOP. Otherwise, return to step 1.

Figure 1.3: Outline of the Branch-and-Cut Algorithm

for subproblem P^i based on x^i_{LP} as F^i. If $F^i \neq \emptyset$, we denote the constraint matrix and the right hand side vector corresponding to the set of valid inequalities in F^i using notation $[D^i, d^i]$ where $D^i \in \mathbb{Q}^{|F^i| \times n}$ and $d^i \in \mathbb{Q}^{|F^i|}$. An outline of a branch-and-cut algorithm is presented in Figure 1.3.

1.2.8 Column Generation Algorithms

Column generation is an approach for solving LPs with a large number of variables for which we want to avoid explicitly enumerating all columns. By solving a sequence of linear programs and dynamically generating columns eligible to enter the basis, such an algorithm implicitly considers

all columns but generates only those that may improve the objective function. The objective of the algorithm is to determine a subset of the variables that includes those that are nonzero in some optimal solution. By solving the restricted version of LP including only this subset of columns rather than all possible columns, we can obtain the optimal LP solution.

Column generation algorithms have been applied to a wide variety of large-scale optimization problems, such as vehicle routing, crew scheduling, cutting stock, lot sizing and machine scheduling. Lübbecke and Desrosiers (2005) list other problems to which column generation has been applied and provide an excellent review of the methodology. To describe the algorithm in more detail, we consider the following LP.

$$\begin{aligned} & \text{Minimize} && \sum_{q \in Q} c_q \theta_q \\ \textbf{(CP)} \qquad & \text{subject to} && \sum_{q \in Q} A_q \theta_q \geq d, \\ & && \theta_q \geq 0 \quad \forall q \in Q, \end{aligned}$$

where Q is the index set for all columns, $A \in \mathbb{Q}^{m \times |Q|}$ is the constraint matrix, $A_q \in \mathbb{Q}^m$ is the vector associated with column q and $d \in \mathbb{Q}^m$ is the right hand side vector. Assume that $|Q|$ is large and thus enumeration of all columns is inefficient. We refer to (CP) as *linear master problem*. In each iteration of the column generation algorithm, a restricted version of (CP) which is referred to as *restricted master problem* (RMP) that includes only a subset of columns is solved. Assuming that the RMP is feasible, by solving RMP, we can obtain a dual solution vector π. Next, the algorithm searches for a column A_q for some $q \in Q$ with the smallest *reduced cost* $\hat{c}_q$, where $\hat{c}_q = c_q - \pi^\top A_q$. This search step is called *pricing*.

In the pricing steps of the algorithm, for a given dual solution $\bar{\pi}$, the following optimization model is solved.

$$\begin{aligned} \hat{z} = \text{Minimize} \quad & c(x) - \bar{\pi}^\top x \\ \textbf{(SP}(\pi)\textbf{)} \qquad \text{subject to} \quad & x \in \{A_q | q \in Q\}, \end{aligned}$$

COLGEN
Input: An instance (CP) with a set $Q_0 \subseteq Q$ of initial columns (containing a feasible solution).
Output: The optimal solution value of (CP).

0. Let $Q_0 \subseteq Q$ be the index set for the initial columns. Set $k \leftarrow 0$.
1. Build RMP$_k$ with the columns in Q_k.
2. Solve RMP$_k$ and obtain π_k.
3. Call SP(π_k).
   a. If $\hat{z}_k < 0$, then set $k \leftarrow k+1$, $Q_k \leftarrow Q_{k-1} \cup \{k\}$. Go to step 1.
   b. Otherwise, the solution for RMP$_k$ is an optimal solution of (CP). STOP.

Figure 1.4: Outline of the Column Generation Algorithm

where $c(x)$ is a function of x, and results in c_q for the corresponding member A_q. (SP(π)) is referred to as the *pricing problem* (or *subproblem*).

The objective of the pricing problem is to either identify new columns to add to the current RMP or conclude that the RMP solution is optimal for (CP). From the theory of linear programming, we know that if every variable has a non-negative reduced cost, then the solution is optimal. If $\hat{z} < 0$ for a given $\bar{\pi}$, then $\hat{x}$ that results in the objective $\hat{z}$ can be added to the RMP. Then, the updated RMP is re-optimized. The algorithm continues iteratively solving the RMP and the pricing problem until $\hat{z} \geq 0$ for a dual vector obtained in an iteration of the algorithm.

Before we outline the steps of a column generation algorithm, which we refer to as COLGEN, we describe some notation. Let Q_k be the set of indices of the columns at k^{th} iteration of the algorithm for $k \geq 0$. To ease the description, let RMP$_k$ be the restricted version of (CP) including columns in set Q_k at iteration k and let π_k be the dual variable vector obtained by solving RMP$_k$ for all $k \geq 0$. Similarly SP(π_k) represents the pricing routine for a given π_k. Let $\hat{z}_k$ and $\hat{x}_k$ be the optimal solution value and an optimal solution vector obtained from SP(π_k), respectively. An outline of a column generation algorithm is presented in Figure 1.4.

1.2.9 Dantzig-Wolfe Decomposition

The Dantzig-Wolfe Decomposition (DWD) approach was developed by Dantzig and Wolfe (1960) to solve large scale linear programs. The methodology aims to obtain a reformulation of the original problem that leads to the iterative solution of many smaller problems instead of the original problem. DWD has become a widely used method in (mixed) integer linear programming to obtain models with stronger LP relaxation bounds. We first explain the steps of the DWD method for linear programs and then extend the review for mixed integer linear programming. Consider the LP:

$$\begin{aligned} \text{Minimize} \quad & c^\top x \\ \text{subject to} \quad & A^1 x \geq d^1, \\ & A^2 x \geq d^2, \\ & x \in \mathbb{R}^n_+, \end{aligned} \tag{1.7}$$

where $A^1 \in \mathbb{Q}^{m_1 \times n}$ and $A^2 \in \mathbb{Q}^{m_2 \times n}$ are constraint matrices, and $d^1 \in \mathbb{Q}^{m_1}$ and $d^2 \in \mathbb{Q}^{m_2}$ are rational right hand side vectors. Let $\mathcal{X} = \{x \in \mathbb{R}^n_+ : A^2 x \geq d^2\}$ be the polyhedron defined by the constraint matrix A^2. In this research, we focus on instances in which $\mathcal{X}$ is bounded. We can describe $\mathcal{X}$ as the convex combination of its the extreme points:

$$\mathcal{X} = \{x \in \mathbb{R}^n_+ | x = \sum_{q \in Q} \mathrm{x}^q \theta_q, \sum_{q \in Q} \theta_q = 1, \theta \in \mathbb{R}^{|Q|}_+\}, \tag{1.8}$$

where $\{\mathrm{x}^q\}_{q\in Q}$ is the set of extreme points of $\mathcal{X}$ with finite index set Q. Based on this alternative definition of $\mathcal{X}$, model (1.7) can be reformulated.

$$\begin{aligned}
\text{Minimize} \quad & c^\top(\sum_{q\in Q}\mathrm{x}^q\theta_q) \\
\text{subject to} \quad & A^1(\sum_{q\in Q}\mathrm{x}^q\theta_q) \;\geq\; d^1, \\
& \sum_{q\in Q}\theta_q \;=\; 1, \\
& \theta \;\in\; \mathbb{R}_+^{|Q|}.
\end{aligned}$$

If we rearrange and let $c_q = c^\top \mathrm{x}^q$ and $A_q = A^1\mathrm{x}^q \;\forall q \in Q$, we obtain the following model:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{q\in Q} c_q\theta_q \\
\text{subject to} \quad & \sum_{q\in Q} A_q\theta_q \;\geq\; d^1, \\
& \sum_{q\in Q}\theta_q \;=\; 1, \\
& \theta \;\in\; \mathbb{R}_+^{|Q|}.
\end{aligned} \tag{1.9}$$

As in the previous section, model (1.9) is referred to as *linear master problem*. To find the solution of problem (1.9), we have to explore the set of extreme points of $\mathcal{X}$, which is possibly large. The COLGEN procedure can be employed to solve problem (1.9).

As we noted before, the DWD methodology can be used to obtain a stronger lower bound for an MILP. Consider the MILP, which is obtained by adding integrality restrictions to model (1.7).

In particular,

$$\begin{aligned} & & \text{Minimize} \quad & c^\top x & \\ \textbf{(D)} \qquad & & \text{subject to} \quad & A^1 x \geq d^1, & (1.10)\\ & & & A^2 x \geq d^2, & (1.11)\\ & & & x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+, & (1.12) \end{aligned}$$

for some integer $n \geq r > 0$. In the context of mixed integer programming, the DWD method aims to get a possibly better lower bound than the LP relaxation of (D) by optimizing the intersection of the polyhedron defined by constraints (1.10) with the convex hull of constraints (1.11) and (1.12). Based on this idea, we are interested in the convex hull of constraints (1.11) and (1.12). Let $X = \{\mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ : A^2 x \geq d^2\}$ be the set of feasible solutions defined by constraints (1.11) and (1.12), and let $conv(X)$ be the convex hull of X. We can express $conv(X)$ using the convex combination of the solutions in X:

$$conv(X) = \{x \in \mathbb{R}^n | x = \sum_{q \in E} \mathrm{x}^q \theta_q, \sum_{q \in E} \theta_q = 1, \theta \in \mathbb{R}^{|E|}_+\}, \tag{1.13}$$

where $\{\mathrm{x}^q\}_{q \in E}$ is the set of points in X and E is the index set. We can reformulate (D) by substituting x based on (1.13). The LP relaxation of this reformulation is as follows:

$$\begin{aligned} \text{Minimize} \quad & \sum_{q \in E} (c^\top \mathrm{x}^q)\theta_q & \\ \text{subject to} \quad & \sum_{q \in E} (A^1 \mathrm{x}^q)\theta_q \geq d^1, & \\ & \sum_{q \in E} \theta_q = 1, & (1.14)\\ & \theta \in \mathbb{R}^{|E|}_+. & \end{aligned}$$

The optimal solution value of problem (1.14) is an upper bound for the LP relaxation of (D). If all

extreme points of $\mathcal{X}$ satisfy integrality restrictions, then the LP relaxation of (D) is equal to the optimal solution of problem (1.14). If we aim to find feasible solutions to (D) using the reformulation (1.14), we also need to satisfy the integrality restrictions on x. Let $\bar{\theta}$ be a solution of problem (1.14). When we transform this into x space, the corresponding solution, $\bar{x} = \sum_{q \in E} \mathrm{x}^q \bar{\theta}_q$, may not satisfy (1.12). In order to satisfy integrality constraints, two modeling approaches are proposed (see Vanderbeck (1994)). The first one is called *discretization* in which integrality restrictions are added for (θ):

$$\begin{aligned} \text{Minimize} \quad & \sum_{q \in E} (c^\top \mathrm{x}^q) \theta_q \\ \text{subject to} \quad & \sum_{q \in E} (A^1 \mathrm{x}^q) \theta_q \geq d^1, \\ & \sum_{q \in E} \theta_q = 1, \\ & \theta \in \mathbb{Z}_+^{|E|}. \end{aligned} \tag{1.15}$$

The second approach is called *convexification* in which the integrality conditions are forced for the variables in the original space (x).

$$\begin{aligned} \text{Minimize} \quad & \sum_{q \in E} (c^\top \mathrm{x}^q) \theta_q \\ \text{subject to} \quad & \sum_{q \in E} (A^1 \mathrm{x}^q) \theta_q \geq d^1, \\ & \sum_{q \in E} \theta_q = 1, \\ & \theta \in \mathbb{R}_+^{|E|}, \\ & x = \sum_{q \in E} \mathrm{x}^q \theta_q, \\ & x \in \mathbb{Z}_+^r \times \mathbb{R}_+^{n-r}. \end{aligned} \tag{1.16}$$

Problems (1.15) and (1.16) are the reformulations of (D) using the DWD procedure and in general

referred to as *integer master problems*. In order to solve problems (1.15) and (1.16), the branch-and-price algorithm is used.

1.2.10 Branch-and-Price Algorithms

Another divide-and-conquer approach that is used to solve MILP models is *branch and price*, which is a combination of the branch-and-bound algorithm and the column generation algorithm. As in the branch-and-bound algorithm, the original feasible solution set is partitioned into subsets and each subset is recursively treated by applying bounding and branching. Similar to an LP-based branch-and-bound algorithm, a lower bound is determined by solving the LP relaxation of each subproblem. However, because of the formulation of the original problem, for each subproblem, column generation is employed to obtain the LP relaxation bound. After the LP relaxation solution is obtained, the branching procedure is applied.

The algorithm has been applied to a variety of problems, including various vehicle routing problems (Fukasawa et al. (2006)), location routing problems (Berger et al. (2007)) and airline crew scheduling problems (Vance et al. (1997)), all of which are relevant in the context of the LRSP. Although branch and price has been applied successfully to various problems, there are many difficulties associated with the algorithm and the success of the algorithm highly depends on the structure of the problem. Barnhart et al. (1998) discuss the general methodology and the many challenges in developing a branch-and-price algorithm.

One major challenge in a branch-and-price algorithm is the efficiency of the column generation algorithm used. Since the column generation algorithm is applied for each subproblem, the efficiency of the column generation procedure has a major effect on the overall performance of the algorithm. Another challenge is the determination of the branching disjunctions. To solve the LP relaxation of each subproblem which is obtained by applying terms of the branching disjunction associated with the subset, we need to use the column generation algorithm that will provide columns feasible for the associated subset of the original feasible solution set. In order to do that,

the pricing problem in the column generation algorithm may need to be modified. These modifications may result in more difficult pricing problems. Therefore, in branch-and-price algorithms, the branching decisions are crucial because of their effect to the pricing problem. They may effect the performance of the overall algorithm significantly.

Next, we present an outline of a branch-and-price algorithm. We use the same notation that is used to describe BRANCH-BOUND and define some additional notation. We define set C_G global to the algorithm to represent the set of columns generated so far. We use notations (RMP$_k^i$) to denote the restricted version of the LP relaxation of P^i at the k^{th} iteration of the column generation algorithm. We let π_k^i represent the dual variable vector obtained by solving RMP$_k^i$ and SP(π_k^i) represent the pricing routine for a given π_k^i. SP(π_k^i) results in the set of generated columns that is denoted by C_k^i. We let x_k^i denote the solution of RMP$_k^i$ and z_i^k be the solution value corresponding to x_k^i. If $C_k^i = \emptyset$, then $z_k^i = z_{LP}^i$. An outline of a branch-and-price algorithm is presented in Figure 1.5.

1.2.11 Branch-Cut-and-Price Algorithms

A branch-cut-and-price algorithm combines the branch-and-price algorithm with a branch-and-cut algorithm. In a fashion similar to the branch-and-cut procedure, valid inequalities are optionally generated for each subproblem to strengthen the LP relaxation bound obtained via column generation algorithm. The cut generation procedure may be done before, after or during the column generation algorithm. However, at any node, after a set of valid inequalities are added to the restricted version of the subproblem, the column generation algorithm must be called again to solve or find a lower bound for the LP relaxation of the restricted version of the subproblem extended with valid inequalities.

As in a branch-and-cut procedure, the decision to employ a given class of valid inequalities depends on factors such as the efficiency with which the separation problem can be solved and the degree of improvement on the LP relaxation bound expected by adding these inequalities. In addition to these factors, in a branch-cut-and-price algorithm, the structure of the valid inequalities

BRANCH-PRICE
Input: An instance (P) with initial column set C^0.
Output: z_{IP} and x_{IP}.

0. Set $z_{UB} \leftarrow \infty$ or to the value of a known feasible solution, $z_{LB} \leftarrow -\infty$, $C_G \leftarrow C^0$ and $L \leftarrow \{\text{P}^0\}$.
1. Select a subproblem P^i from L. Set $L \leftarrow L \backslash \{\text{P}^i\}$.
2. Set $k \leftarrow 0$.
3. Build RMP^i_k with the columns in C_G and solve. Call SP(π^i_k). Set $C_G \leftarrow C_G \cup C^i_k$.
   (a) If $C^i_k = \emptyset$, set $z^i_{LP} \leftarrow z^i_k$.
   - If $z^i_{LP} \geq z_{UB}$, go to step 6.
   - If $x^i_{LP} \in S$, then set $z_{UB} \leftarrow \min\{z^i_{LP}, z_{UB}\}$ and go to step 6.
   - Otherwise, go to step 4.

   (b) Set $k \leftarrow k + 1$ and return to step 3.
4. Branch (partition S^i into subsets). Define subproblems associated with subsets. Add new subproblems to L.
5. $z_{LB} = \min\{z^k_{LP} | \text{P}^k$ is a parent node for problem $\text{P}^i \in L\}$.
6. If $z_{LB} \geq z_{UB}$ or $L = \emptyset$, then set $z_{IP} \leftarrow z_{UB}$ (if there is a solution vector that yields z_{UB}, then it is x_{IP}), and STOP. Otherwise, return to step 1.

Figure 1.5: Outline of the Branch-and-Price Algorithm

may also be important, since, as with the branching decisions, they may affect the structure of the pricing problem. If generation of a class of valid inequalities significantly affects the efficiency of the pricing algorithm, employment of these inequalities may not be desirable, as the gain from the lower bound improvement may not be offset by the overall performance decline because of the modifications to the pricing algorithm.

In order to outline a branch-cut-and-price algorithm, we use the notation defined before for BRANCH-BOUND and BRANCH-PRICE, and define some additional notation. We denote the set of valid inequalities generated for subproblem P^i based on x^i_k as F^i_k. If $F^i_k \neq \emptyset$, we denote the constraint matrix and the right hand side vector corresponding to the set of valid inequalities in F^i_k

using notation $[D_k^i, d_k^i]$ where $D_k^i \in \mathbb{Q}^{|F_k^i| \times n}$ and $d_k^i \in \mathbb{Q}^{|F_k^i|}$. An outline of a branch-cut-and-price algorithm is presented in Figure 1.6.

1.3 Related Work

The LRSP integrates facility location, vehicle routing and vehicle scheduling decisions. In this section, we briefly review the literature for each component problem as well as the literature for related integrated problems.

1.3.1 Facility Location

The first component of the LRSP is the question of where to locate the facilities and how to allocate the customers to these facilities. Determining where to locate facilities is an important problem that arises in the design of distribution networks. ReVelle and Eiselt (2005) and Klose and Drexl (2005) provide recent surveys of the various types of location problems, formulations and solution algorithms. The facility location literature encompasses a wide variety of problems, which can be categorized in a variety of ways. Daskin (1995) provides a taxonomy of facility location problems and describes methods for categorizing location problems. Location problems can be grouped as planar, network, or discrete problems, based on the allowable location for the facilities and the customers. Problems in which customers and facilities can be located anywhere on a plane are planar location problems. Problems in which the customers and the candidate locations are restricted to a discrete set are discrete location problems. Network location problems are a special case of discrete location problems in which the candidate facilities and the demand locations are restricted to the nodes of a network and travel is only possible via the arcs of network.

Most relevant to our study of the LRSP are discrete location problems. Labbé and Louveaux (1997) provide a survey of the literature related to discrete facility location problems including application areas, solution methods and model extensions. Early work on discrete location problems started with Hakimi (1964, 1965), who introduced the p-median problem on a network, which is

BRANCH-CUT-PRICE
Input: An instance (P) with initial column set C^0.
Output: z_{IP} and x_{IP}.

0. Set $z_{UB} \leftarrow \infty$ or to the value of a known feasible solution, $z_{LB} \leftarrow -\infty$, $C_G \leftarrow C^0$, and $L \leftarrow \{\text{P}^0\}$.
1. Select a subproblem P^i from L. Set $L \leftarrow L \backslash \{\text{P}^i\}$.
2. Set $k \leftarrow 0$.
3. Build RMP_k^i with the columns in C_G and solve. Call SP(π_k^i). Set $C_G \leftarrow C_G \cup C_k^i$.
   (a) If $C_k^i = \emptyset$, set $z_{LP}^i \leftarrow z_k^i$.
   - If $z_{LP}^i \geq z_{UB}$, go to step 7.
   - If $x_{LP}^i \in S$, then set $z_{UB} \leftarrow \min\{z_{LP}^i, z_{UB}\}$ and go to step 7.
   - Otherwise, go to next step.

   (b) If cut generation is desired, go to step 4.
   (c) If $C_k^i = \emptyset$ go to step 5. Otherwise, set $k \leftarrow k + 1$ and return to step 3.
4. Generate valid inequalities violated by x_k^i. Let F_k^i be the set of inequalities generated.
   - If $F_k^i \neq \emptyset$, set $\mathcal{P}^i \leftarrow \mathcal{P}^i \cap \{x \in \mathbb{R}^n | D_k^i x \geq d_k^i\}$. Return to step 2.
   - If $C_k^i \neq \emptyset$, return to step 3.
   - Otherwise, proceed.
5. Branch (partition S^i into subsets). Define subproblems associated with subsets. Add new subproblems to L.
6. $z_{LB} = \min\{z_{LP}^k | \text{P}^k$ is a parent node for problem $\text{P}^i \in L\}$.
7. If $z_{LB} \geq z_{UB}$ or $L = \emptyset$, then set $z_{IP} \leftarrow z_{UB}$ (if there is a solution vector that yields z_{UB}, then it is x_{IP}), and STOP. Otherwise, return to step 1.

Figure 1.6: Outline of the Branch-Cut-and-Price Algorithm

known to be NP-hard (Kariv and Hakimi (1979)). The objective is to locate p facilities on a network with demand nodes so as to minimize the total weighted distance between the customers and the closest facilities to them. ReVelle and Swain (1970) were the first to formulate the p-median problem as a 0-1 integer program. Since then, researchers have developed both exact methods (e.g., Galvao (1993), Bilde and Krarup (1977)) and heuristic methods (e.g., Maranzana (1964), Teitz and Bart (1968), Rosing and ReVelle (1997)) for the p-median problem. Depending on the difficulty of the instances, optimal or near optimal solutions for p-median problems are reported for instances up to 3000 customer nodes.

By removing the constraint that p locations must be selected and by adding a fixed cost for each candidate location, we obtain a simple plant location problem from the p-median problem. Also called the uncapacitated facility location problem (UFLP), the problem has been well-studied in the literature (Cornuejols et al. (1990)). The objective of the problem is to select a subset of the candidate facilities and to assign customers to the selected facilities so as to minimize the sum of fixed and operating costs. The operating cost in the UFLP is the transportation cost calculated assuming individual customer routes that originate at a facility, serve one customer and return to the same facility. The problem is NP-hard (Cornuejols et al. (1990)). A number of researchers have focused on polyhedral analysis of the UFLP. Although a complete description of the UFLP polytope is not known, Guignard (1980), Cornuejols and Thizy (1982) and Cho et al. (1983 a,b) provide partial descriptions. Others have focused on developing algorithms for the UFLP, for example, branch-and-bound based algorithms by Erlenkotter (1978) and Körkel (1989), a Lagrangian relaxation algorithm by Cornuejols et al. (1977), and a dual simplex algorithm by Simao and Thizy (1989). The best results for the problem are reported by Körkel (1989) using an approach that extends the dual-ascent heuristic procedure developed by Erlenkotter (1978) (ReVelle and Eiselt (2005)).

By adding capacity constraints on the facilities, we obtain the capacitated facility location problem (CFLP) from the UFLP. Leung and Magnanti (1989) and Aardal et al. (1995) investigate the polyhedral structure of the CFLP and introduce facet defining inequalities for the problem,

which lead to cutting plane algorithms (Aardal (1998)) for solving the problem. By using an extended formulation (with other valid inequalities) of the CFLP, Cornuejols and Thizy (1982) find and compare lower bounds for the CFLP by relaxing different constraints within a Lagrangian relaxation and decomposition algorithm. Other papers presenting various solution algorithms for the CFLP are Baker (1986), Roy (1986), Pirkul (1987) and Beasley (1988).

In the CFLP, the demand of a customer can be supplied by more than one facility. However, in some applications, there may be a single-source requirement for a customer, which means that the total demand of a customer must be supplied by exactly one facility. When the single source requirement is added to the CFLP, it is called the capacitated facility location problem with single sourcing (CFLPSS). The CFLPSS is more difficult than the CFLP since, while the CFLP is a mixed integer model, the CFLPSS is a pure integer model with same number of variables and constraints. In addition, for a given set of open facilities, the CFLP solves a transportation problem to find the assignment of customers to the facilities, but the CFLPSS solves a generalized assignment problem, which is also an NP-hard problem. In the literature, most of the studies use heuristics, especially Lagrangian based heuristics, such as Klincewicz and Luss (1986) and Sridharan (1993). Delmaire et al. (1997) design several metaheuristics and Ahuja et al. (2004) develop a neighborhood search algorithm for the problem. Holmberg et al. (1999) develop an exact branch-and-bound algorithm that uses a Lagrangian based heuristic and repeated matching heuristic that solves successive matching problems. They can solve problems up to 200 customers and 30 facilities.

1.3.2 Vehicle Routing

The second component of the LRSP is the decision of how to design the vehicle routes. The classical vehicle routing problem (VRP) finds a set of minimum cost vehicle routes, each of which starts at the depot, visits some customer locations and returns to the depot. There is a large body of literature related to the VRP that describes the many problem variations and solution approaches. The VRP has many variations depending on the characteristics of the vehicles, the facilities and

the customers. For example, the vehicles may be either identical or differ with respect to capacity limits; the problem may be concerned with delivery only, pick-up only, or both delivery and pick-up; the vehicles might be restricted to serve each customer in a given time window and/or serve according to precedence relations of the customers. In addition, the problem may involve a single facility or multiple facilities. Christofides (1985) surveys the different types of formulations used to model the VRP and the exact and heuristic algorithms designed for the problem prior to 1985. In a recent study, Toth and Vigo (2002) give a detailed description of the variants of the VRP and their formulations, and they discuss the solution methods that are applied to these variants.

To formulate the VRP as an integer program, three different approaches are used in the literature. One approach is to define integer variables for each arc or edge in the model representing the number of times the arc or edge is traveled. This type of model, which is called a *vehicle flow model*, can be divided into two groups: *two index formulations* and *three index formulations* (Laporte and Nobert (1987)). In a two index formulation, a flow variable has two indices representing the origin and the destination of the arc and in a three index formulation, each variable has an additional index representing the vehicle traveling the arc. Vehicle flow models, especially two index formulations, are very common in the VRP literature.

A classical vehicle flow model includes constraints called generalized subtour elimination constraints, the number of which grows exponentially with the number of nodes in the problem. There is one subtour elimination constraint for each subset of customers, which forces a lower bound for the number of vehicles exiting the subset based on the minimum number of vehicles necessary to serve the customers in the subset. The exponential number of constraints is handled by considering a relaxation of the model that includes only some of the constraints or by using a branch-and-cut algorithm. Miller, Tucker, and Zemlin (1960) introduce a polynomial number of constraints that eliminate subtours, but the LP relaxation of the model with these new constraints is weaker than the model with generalized subtour elimination constraints. Desrochers and Laporte (1991) strengthen the Miller-Tucker-Zemlin subtour elimination constraints by lifting them. They define stronger valid inequalities that can be adapted to variations of the VRP, such as the

capacitated VRP (CVRP), the distance constrained VRP (DVRP) and the VRP with time windows (VRPTW).

A second modeling approach, developed by Garvin et al. (1957), is called a *commodity flow model*. In this approach, an additional variable for each arc is added to the variables in the vehicle flow model. These additional variables represent the amount of demand that flows along the edges traveled by the vehicles. Letchford and González (2006) analyze three different commodity flow models: one commodity, two commodity and multi commodity flow models. In two commodity flow model, in addition to the flow on each arc representing the vehicle load, there is a reverse flow equal to the vehicle residual capacity. In multi commodity flow model, demand of each customer represents one commodity and flow of each commodity is denoted by different variables on each arc. No exact solution algorithms for this approach are reported in the literature. Based on the computational experiments, Gavish and Graves (1979) report large gaps between the LP solutions and the optimal integer solutions when the number of customers are greater than 15.

A third approach, proposed by Balinski and Quandt (1964), is a set partitioning formulation, in which there is one variable corresponding to each feasible vehicle route. The model has possibly exponentially large number of binary variables representing whether a specific route is selected or not. Since the feasibility of each route is considered in the process of constructing the set of routes to be included in the model, various routing constraints, such as vehicle capacity, route length restrictions and time windows for the customer nodes, can be included in the model without changing the formulation. Because of the large number of variables, it is usually solved by a column generation algorithm. The LP relaxation of the model is generally tight (Toth and Vigo (2002)).

Letchford and González (2006) derive theoretical relationships between these different formulations described in previous paragraphs. Using projection, they compare the LP relaxation bounds. For capacitated VRP (CVRP), they prove that the three index formulation strengthened with certain classes of valid inequalities gives the same lower bound with a two index formulation

with simpler constraints. Therefore, the LP relaxation bound based on the three index formulation is weaker than the LP relaxation bound based on the two index formulation. They provide some results for the relationship between a commodity flow model and a two index vehicle flow model, and they show that some valid inequalities in the two index space are implied by extra flow variables in commodity flow formulations. In addition, Letchford and González (2006) prove the strength of the set partitioning formulation by showing that the set partitioning variables imply some of the valid inequalities in the two index space. Another study investigating relationships between formulations is Gouveia (1995) which includes some theoretical results that compare commodity flow models and two index vehicle flow models.

In reviewing the solution algorithms for the VRP, we highlight those for deterministic, capacitated VRPs. In a capacitated VRP (CVRP), the total demand of a route must be less than the capacity of the vehicle. Both exact methods and heuristic methods have been developed to solve the CVRP. Laporte and Nobert (1987) survey the exact algorithms for solving the VRP. They give a complete analysis of branch-and-bound algorithms, which are the most common way to solve the CVRP. They also describe dynamic programming and integer programming approaches that use different formulations of the problem. In addition to branch-and-bound procedures, Laporte (1992) describes set partitioning formulations, column generation procedures and heuristic approaches (e.g. Clark and Wright, tabu search and sweep algorithm) for the problem.

Branch-and-cut algorithms are very successful in finding optimal solutions for large instances of the symmetric TSP, which is a problem closely related to the CVRP. This success has motivated the development of branch-and-cut algorithms for vehicle flow models of the CVRP. Toth and Vigo (2002) describe how to apply a branch-and-cut algorithm to the VRP and the CVRP. Branch-and-cut algorithms incorporate the results of polyhedral studies of the CVRP polytope, i.e., valid inequalities (e.g. Laporte and Nobert (1981b) and Laporte et al. (1985)) and facets of the polytope. The first papers published about the facets of the CVRP polytope were by Araque, Hall, and Magnanti (1990) and Araque (1990), who assumed unit demand for customers, and by Cornuejols and Harche (1993), who extended the research to general demand. One of the main difficulties

with branch-and-cut algorithms is that the separation procedure for the cuts might be NP-hard. To overcome this difficulty, researchers seek polynomial time algorithms for exact separation (e.g., McCormick et al. (2002) and Blasum and Hochstättler (2000)) and heuristic separation algorithms (e.g., Augerat et al. (1999), Ralphs et al. (2003)).

Set partitioning-based formulations are used successfully by Cullen et al. (1981) to solve the VRP heuristically and by Agarwal et al. (1989), Bixby (1999), Hadjiconstantinou et al. (1995) and Desrochers et al. (1992) to solve the VRP exactly. The first step for solving a set partitioning-based formulation is to solve the LP relaxation of the model using column generation. The objective of the column generation subproblem (pricing problem) is to construct feasible vehicle routes which include each customer at most once (elementary routes) and satisfy some other constraints required for the feasibility of a route such as vehicle capacity. Since the complexity of the problem of finding elementary routes is NP-hard, it is common to relax this property and solve the set partitioning formulation with non elementary routes.

In set partitioning based models, to obtain the LP relaxation bound, people use various approaches to solve the column generation subproblem. Agarwal et al. (1989) use a branch-and-bound algorithm, Desrochers et al. (1992) use a branch-and-bound algorithm and dynamic programming, Bixby et al. (1997) use a cutting plane algorithm, and Hadjiconstantinou et al. (1995) use a dynamic programming heuristic and solve the dual of column generation problem. For many of the problems, the LP relaxation solution is very close to integer solution. However, even if the LP relaxation is very strong, some branching may be required to obtain an integer solution. Thus, branch-and-price algorithms (see Desrochers et al. (1992)) are used to obtain integer solutions. To improve the overall performance, branch-and-price algorithm is extended with cutting planes (Fukasawa et al. (2006) and Jepsen et al. (2008)).

A generalization of the VRP, which is also a special case of the LRSP, is the Multi-Depot Vehicle Routing Problem (MDVRP). The MDVRP constructs a set of vehicle routes with minimum cost in which each customer is visited by exactly one vehicle, and each vehicle route starts and ends at the same depot. Exact solution algorithms are very rare for MDVRP; Laporte et al.

(1984) and Laporte et al. (1988) are the only two papers that report exact solution algorithms for the MDVRP. Both solution methods are based on branch-and-bound algorithms. Many papers develop heuristic algorithms, some of them are Tillman (1969), Tillman and Cain (1972), Gillett and Johnson (1976), Chao et al. (1993), Cordeau et al. (1997) and Renaud et al. (1996).

1.3.3 Integrated Facility Location and Routing

In most facility location problems, the implicit assumption is that customers are served on individual routes or that customers travel to the facilities. For situations in which customers receive service from routes making multiple stops, this assumption does not accurately capture the transportation cost. In order to accurately represent the cost of serving customers, we need to know the order of the customers on the route. Hence, the location and routing problems must be solved simultaneously. As early as 1968, Webb (1968) compared using a cost function considering multiple stop tours and a simple cost function based on straight line trips to customers to determine the locations of depots with minimum cost. He concluded that the solutions correspond to minimums of the cost functions did not coincide, and the solution based on a cost function that considers multiple-customer trips could be effected with the changes in the trip restrictions. Salhi and Rand (1989) also investigated the effect of ignoring the routing of vehicles while deciding the locations of facilities. They showed that solving the location and routing problems independently does not always result in the best solution in terms of cost. Thus, although solving the combined problem may be more difficult than solving the individual ones, it may result in a more cost efficient solution.

Laporte (1988) reviews the work on deterministic location routing problems; he describes the different formulations of the problem, the solution methods used and the computational results published up to 1988. More recent review papers by Min et al. (1998) and Nagy and Salhi (2007) classify both deterministic and stochastic problems in the LRP literature with respect to problem characteristics and solution methods. They group the LRP papers according to characteristics

such as single- or multi-stage, deterministic or stochastic, single or multiple facilities and vehicles, uncapacitated or capacitated vehicles and facilities and solution space (discrete, network or continuous). Because the LRP combines two NP-hard problems, researchers have developed a wide variety of both heuristic and exact algorithms for the LRP. Most of the heuristic algorithms tend to divide the problem into subproblems. Heuristic algorithms handle these subproblems sequentially or iteratively. Based on Min (1996), for problems with large facility fixed costs and capacitated vehicles, sequential approaches perform computationally better. Some examples of sequential heuristic algorithms are developed by Perl and Daskin (1985), Hansen et al. (1994) and Barreto et al. (2007).

Perl and Daskin (1985) develop a heuristic for a three index flow formulation of an LRP with capacitated distribution centers and vehicles. They decompose the problem into three subproblems: a multi-depot vehicle dispatch problem, a warehouse location-allocation problem and a multi-depot routing allocation problem. They develop an algorithm that solves the problems sequentially but considers the dependency between the problems. They solve the multi-depot vehicle dispatch problem and the multi-depot routing allocation problem heuristically and the location-allocation problem exactly. Hansen et al. (1994) introduce flow variables and flow constraints to the model in Perl and Daskin (1985) to get an MILP and solve the new model with a heuristic that is a modification of the heuristic developed by Perl and Daskin (1985). Barreto et al. (2007) develop a sequential heuristic algorithm for problems with capacitated facilities and capacitated vehicles. In their heuristic, the first step is to solve routing subproblem. They group customers using clustering techniques and design TSP solutions. Then, by assuming each group of customers as a single customer, single source capacitated location routing problem is solved.

Srivastava (1993) presents 3 iterative heuristics called SAV1, SAV2 and CLUST and compares the results with a classical sequential heuristic in which facility locations are determined first and vehicle routes are designed based on open facilities. SAV1 heuristic starts with all depots being open and using a multi depot vehicle routing heuristic calculates approximate routing costs for the facilities. At each step a facility is closed until the problem becomes infeasible. SAV2 heuristic

works similar to SAV1 except initially all depots are closed and at each iteration one depot is opened. CLUST heuristic generates groups of customers using a minimum spanning tree and a density search clustering technique. Then a depot is selected for each cluster considering the distance between cluster centroid and the depot.

Albareda-Sambola et al. (2005) provide a mathematical model that is constructed based on a network representing an LRP with a single uncapacitated vehicle at each possible depot location. They construct an initial solution by applying a rounding procedure to the LP relaxation of the model, and develop a tabu search algorithm that starts with this initial solution. They develop an additional lower bound by solving an asymmetric traveling salesman problem (ASTP) that includes all depots and customers. Then, they combine the cost of ASTP and a knapsack problem that determines a lower bound for the total depot costs and the cost of connecting depots to customers. Even though the obtained lower bound is generally better than the LP relaxation bound of their LRP model, they conclude that the bound is not tight enough, and they report large gaps between this lower bound and, both the tabu search upper bound and the optimal solution. In addition, by comparing their tabu search upper bound with an upper bound that is derived via a sequential heuristic, they restate the strength of integrated approaches.

Prins et al. (2007) develop a heuristic approach that combines Lagrangean Relaxation and granular tabu search (Toth and Vigo (2003)) for the LRP with capacitated facilities and capacitated vehicles. They divide the algorithm into two phases and alternate between each phase. In the first phase, for a given solution, they aggregate the routes into a single customer and solve a location problem using a Lagrangean relaxation. In the second phase, based on the solution of the location problem, they construct new routes by solving a MDVRP using the granular tabu search heuristic. They evaluate their algorithm using randomly generated instances up to 200 customers.

Exact algorithms for the LRP have been developed mostly for two index vehicle flow formulations. Laporte and Nobert (1981a) solve a single depot model with a constraint relaxation method and a branch-and-bound algorithm. They report solving problems with 50 customer locations. Laporte et al. (1983) solve a multi-depot model using a constraint relaxation method and Gomory

cutting planes to satisfy integrality. They can solve problems with at most 40 customer sites. Laporte et al. (1986) apply a branch-and-cut algorithm to a multi-depot LRP model with vehicle capacity constraints. They use subtour elimination constraints and chain barring constraints that guarantee that each route starts and ends at same facility. The authors report computational results for problems with 20 customer locations and 8 depots.

Belenguer et al. (2006) provide two formulations of the LRP with capacitated facilities and capacitated vehicles using two-index variables. They present a set of valid inequalities for the problem and develop two branch-and-cut algorithms based on the different formulations of the problem. They report that they solve instances with up to 32 customers to optimality in less than 70 CPU seconds and provide good lower bounds for the rest of the instances, which have up to 134 customers.

Berger (1997) develops a set partitioning-based model for an LRP with route-length constraints and uncapacitated vehicles and facilities. Berger et al. (2007) extend the work to develop an exact branch-and-price algorithm in which they solve the pricing problem as an elementary shortest path problem with one resource constraint. They report computational results for problems with 100 customers and various distance constraints.

1.3.4 Vehicle Scheduling

The third component of the LRSP is how to assign routes to vehicles, which is known as the vehicle scheduling or loading problem. The (multi-depot) vehicle scheduling problem ((MD)VSP) seeks a least-cost assignment of routes (or trips), each with a starting and ending time, to vehicles such that each route is assigned to one vehicle, each vehicle starts a route after the starting time and finishes before the ending time and each vehicle starts from a depot and returns to same depot. In some variations, there are constraints such as depot capacity and upper and lower bounds on the number of vehicles at each depot. The MDVSP is a special case of the LRSP; given a set of selected facilities and a set of vehicle routes connected to these facilities, the LRSP reduces to a MDVSP.

Carraresi and Gallo (1984) describe how to convert a single depot VSP to an assignment problem or a minimum cost network flow model and therefore they conclude that the single depot VSP is easy to solve. However, Bertossi et al. (1987) show that if there are more than 2 depots, the MDVSP is NP-hard.

Both heuristic and exact solution methods are proposed for the MDVSP. Bodin and Golden (1981) propose two heuristics based on decomposing the problem into single depot VSPs. Bertossi et al. (1987) model the MDVSP by defining an independent problem for each depot and additional constraints to cover each trip exactly once. They relax the additional constraints using a Lagrangian relaxation algorithm and then solve independent VSPs. Dell'Amico et al. (1993) propose a heuristic method that guarantees a solution with the minimum number of vehicles. They solve shortest path problems on a graph that includes one node for each route and arcs between two nodes that can be assigned to the same vehicle. A path on the graph represents a vehicle duty.

Several exact algorithms based on different formulations of the problem are proposed. Carpaneto et al. (1989) use a branch-and-bound algorithm based on additive lower bounds. Forbes et al. (1994) develop an algorithm using a dual simplex method to solve their three-index integer programming model, which is a multi-commodity network flow problem. Ribeiro and Soumis (1994) and Bianco et al. (1994) use set partitioning-based formulations for which they generate columns and apply a branch-and-bound algorithm to get an integer solution. Fischetti et al. (2001) design a branch-and-cut algorithm by defining valid inequalities of the MDVSP.

1.3.5 Vehicle Routing and Scheduling

In the VRP, there is one-to-one relationship between vehicles and routes, which means that one route is assigned to each vehicle. In a combined vehicle routing and scheduling problem (VRSP), scheduling constraints are added to the VRP constraints to change this one-to-one relationship between vehicles and routes. Therefore, the VRSP seek to determine the set of vehicle routes as well as the assignment of these routes to the available vehicles. In the literature, this version of the VRSP is sometimes called the VRP with multiple trips (VRPMT). Only a few papers in the

literature address the VRPMT and they only describe heuristic approaches.

Taillard et al. (1996) use a three-step algorithm to find good feasible solutions for the VRSP. In the first step, they generate some number of VRP solutions using a tabu search algorithm. In the second step, they select a subset of these routes using an enumerative algorithm and, in the third step, they use a bin packing heuristic to combine the routes in such a way that they will not violate the time constraint and assign them to vehicles. The authors generate test problems with 50 to 199 customers. Feasible solutions are provided for 82 instances out of 104 instances. They report that the cost of the feasible solutions are on the average within 1.2% of the VRP solution obtained by Rochat and Taillard (1995).

Brandao and Mercer (1997) use a tabu search algorithm and the GENI algorithm (developed by Gendreau et al. (1992) to solve TSP) to solve the VRPMT with time window constraints. They design a three-phase algorithm in which the solution is improved in each phase. The input for the first phase is an initial set of routes constructed using VRP constraints. The first phase checks each route in the initial set for the time window infeasibility and, if there is an infeasible route, the route is divided into smaller routes that satisfy the time windows. The second phase constructs a solution with smaller cost by searching the set of routes. Similar to the second phase, the third phase also seeks a solution with lower cost but the solution space is restricted to the feasible space during the search. They evaluate their algorithm by comparing their solutions with manual solutions in a case study. They report on average 20% better solutions. In addition, Brandao and Mercer (1998) solve the instances created by Taillard et al. (1996). They can find feasible solutions for 89 instances out of 104 instances.

Petch and Salhi (2004) develop a multi-stage heuristic algorithm in which they create vehicle routes and then combine the routes using bin packing heuristics. Olivera and Viera (2007) solve the VRPMT using an adaptive memory algorithm that is an enhancement of tabu search. The basic idea is to construct new solutions using the components of the previously generated solutions. They initialize the memory by generating some feasible solutions to the VRP, which they use to construct feasible assignments of routes to vehicles. Then, they use tabu search to improve the

constructed solution. This heuristic differs from the others in that it constructs feasible vehicle assignments in each iteration, while the others only construct the assignment at the final stage. They can find feasible solutions for 95 instances out of 104 Taillard et al. (1996)'s instances. The feasible solutions are on the average within 1.6% of the best known VRP solutions. In infeasible solutions, in which overtime is used, the solutions provided by Olivera and Viera (2007) are better than the solutions provided by other authors.

1.3.6 Integrated Location, Routing and Scheduling

Integrating facility location, vehicle routing and vehicle scheduling decisions gives rise to a class of problems called *integrated location routing and scheduling* problems (LRSP). As far as we know, the papers by Lin et al. (2002) and Lin and Kwok (2006) are the only LRSP papers to appear in the literature. Lin et al. (2002) describe an LRSP, which arises in the delivery of telephone bills, in which they seek to locate facilities and to route and assign vehicles so as to minimize cost. To solve the problem, they develop a heuristic that divides the problem into three phases, facility location, routing and loading, and that iterates through the phases. The location phase determines the set of facilities via enumeration. For a given set of opened facilities, the routing phase constructs vehicle routes using the Clarke and Wright algorithm (Clarke and Wright (1964)) followed by an insertion heuristic and more extensive metaheuristics. The loading phase optimally assigns routes to vehicles considering the time limits. The authors report the development of a branch-and-bound algorithm to solve the exact problem to evaluate their heuristics but they do not provide a formulation. They test their algorithm on five instances with 10-12 customers and 4 potential depots, but the branch-and-bound algorithm can solve only three of the five instances to optimality within the specified time limit. For these instances, they report that their heuristic algorithm can find optimal or good solutions quickly. In addition, they solve a real life instance with 27 customers and 4 depots using their heuristic algorithm and compare the results with the management's manual solution.

Lin and Kwok (2006) extend the study by Lin et al. (2002) to solve a multi-objective LRSP that considers workload balance in terms of total vehicle time and total vehicle load as well as total cost. They use the same three-phase heuristic algorithm designed by Lin et al. (2002). Since they consider multi-objective problems, for each feasible solution in the algorithm, they keep three performance measures, the total cost, the difference between the maximum and the minimum vehicle time and the difference between maximum and minimum vehicle load. The feasible solutions are evaluated based on these three performance measures and only non-dominated solutions are kept. In addition, they design a tabu search (TS) algorithm for the routing phase of the algorithm and compare its performance with the simulated annealing (SA) algorithm applied in Lin et al. (2002). While the TS algorithm and the SA algorithm have similar performance in some instances, the SA algorithm is better than the TS algorithm when the number of demand nodes in a route increases. Furthermore, they compare the performance of the algorithm in two approaches: assigning vehicles to routes after the routing phase (sequential approach) and assigning vehicles while constructing routes (multiple approach). They conclude that the performance of the approaches depends on the difficulty of the instance. The multiple approach is better than the sequential approach when the solution includes routes with more demand nodes. They test their algorithm with two sets of real data, one with 27 customers and one with 89 customers, and a set of simulated data including instances with 100 and 200 customers.

1.4 Contributions of the Thesis

We summarize our contributions as follows.

- We are the first to consider the LRSP formally as a class of problems and to provide mathematical programming models.
- We show that the Dantzig-Wolfe reformulation of the graph-based formulation is equivalent to the set partitioning-based formulation of the problem.

- We show that some valid inequalities defined for the set partitioning-based formulation are not dominated by the original inequalities.

- We are the first to provide an exact solution approach for this class of problems. We develop an exact branch-and-price algorithm. We describe a two-phase solution approach for the pricing problem based on the algorithms available in the literature. We define a set of problem-specific branching disjunctions which guarantees to provide valid partitions of the original feasible solution set and an optimal solution to the problem.

- We extend the branch-and-price algorithm to a two-stage branch-and-price algorithm using the idea of restart and a heuristic extension of the algorithm.

- We evaluate the effect of the inequalities valid for polytopes that are relaxations of the LRSP polytope and its reformulations.

- We show that the chain barring inequalities that are used in the graph-based formulations of the LRP are satisfied by the LP relaxation of our set-partitioning based formulation for the LRSP.

- We develop a methodology to derive disjunctive inequalities in a branch-and-price algorithm and provide examples of various implementations of the methodology for the LRSP.

- We do extensive computational study to demonstrate the effectiveness of the solution approaches we propose and the value of integrating location, routing and scheduling decisions.

- We contribute to the LRP literature by providing an exact branch-and-price algorithm for the problem with capacitated facilities and vehicles. We are not aware of any publication in the literature solving the capacitated version of the problem using a branch-and-price algorithm.

1.5 Outline of the Thesis

We summarize the contents of the remaining chapters as follows. In Chapter 2, we define the LRSP considered in this thesis and present two formulations of the problem: a graph-based and a set partitioning-based. We describe the application of the Dantzig-Wolfe decomposition methodology to one formulation of the LRSP. We describe some valid inequalities to strengthen one formulation of the LRSP.

In Chapter 3, we describe our solution methodology for the LRSP, which is an exact branch-and-price algorithm. We describe the two main components of the algorithm: the pricing problem and the branching disjunctions. We present some implementations used to improve the overall performance of the solution algorithm.

In Chapter 4, we investigate classes of inequalities valid for the LRSP and discuss how we can extend the branch-and-price algorithm to a branch-cut-and-price algorithm. We discuss the adaptation of the disjunctive procedure to generate valid inequalities in a branch-and-price algorithm.

In Chapter 5, we present computational experiments for instances up to 40 customers evaluating the performance of the algorithms described in the previous chapters. In addition, we empirically compare the LRSP solutions with those of some variations of the sequential optimization of facility location, vehicle routing and vehicle scheduling decisions.

In Chapter 6, we discuss solution of the location and routing problem (LRP) under capacity restrictions using a modified version of the algorithm described for the LRSP. We report on our computational experience solving both randomly generated instances and instances from the literature.

In Chapter 7, we summarize the conclusions of the thesis and conclude the thesis with future directions.

Chapter 2

Models and Complexity

In this chapter, we provide a formal description of the location, routing and scheduling problem (LRSP) considered in this research and present two formulations of the problem, one using the commodity flow paradigm and the other using a set partitioning-based approach. We then show that the two formulations are related in that one can be obtained by applying DWD to the other. Finally, we discuss the complexity of the LRSP.

2.1 Problem Definition

The LRSP integrates the decision-making process for determining the optimal number and locations of facilities to be open; an optimal assignment of customers to facilities; an optimal set of vehicle routes from facilities to customers; and an optimal assignment of routes to vehicles subject to scheduling constraints. The objective of the problem is to minimize the total fixed and operating costs associated with facilities and vehicles. We consider an LRSP with capacity constraints on the facilities and on the vehicles, as well as time constraints on the vehicles. In particular, given a set of candidate facility locations and a set of customer locations, we seek to determine a subset of facilities to be open and a routing of vehicles to customers from open facilities in such a way that

a. each customer is visited exactly once,

b. each customer is assigned to exactly one facility,

c. each route starts and ends at the same facility,

d. the total demand of the customers assigned to a route is at most the vehicle capacity,

e. the total working time of a vehicle is no more than the time limit, and

f. the total demand of the customers assigned to a facility does not exceed the capacity of the facility.

In the context of this research, a *route* is a path that starts at a facility and ends at the same facility after visiting at least one customer. A route may only visit customer nodes with the exception of the origin and destination stops, which are at a facility. A route may visit a given customer at most once. Figure 2.1 displays examples of routes. The LRSP is to construct routes originating at open facilities, and also to determine the schedules for vehicles by assigning groups of constructed routes to individual vehicles.

Pairings. To define a schedule for a single vehicle, we adapt the *pairing* concept from the crew scheduling literature (e.g., Desrosiers et al. (1991), Vance et al. (1997), and Anbil et al. (1998)). In the crew scheduling literature, a pairing is a sequence of duty periods and represents a possible schedule for a single crew. In the LRSP context, a pairing is a sequence of routes that represents a possible (feasible) schedule for a single vehicle. Based on the constraints considered here, a pairing is feasible if

a. each route included in the pairing starts and ends at the same facility,

b. for each route in the pairing, the total demand of the customers assigned to the route is at most the vehicle capacity,

c. the total travel time of the pairing is at most the vehicle time limit,

d. each customer included in the pairing is visited only once, i.e., included on only one route.

Figure 2.1 shows two possible pairing constructions. Pairing 1 includes routes 1, 2 and 3; pairing 2 includes routes 4 and 5. Each pairing specifies the routes covered by a single vehicle and has an associated cost, which is defined as the sum of the costs of the component routes plus a fixed cost for the vehicle. Since the cost of a multiple-stop route depends on the sequence of nodes visited, the pairing information implicitly includes sequencing information for each route. In the absence of time windows, the cost of a pairing, however, is not affected by the ordering of the routes and thus is arbitrary.

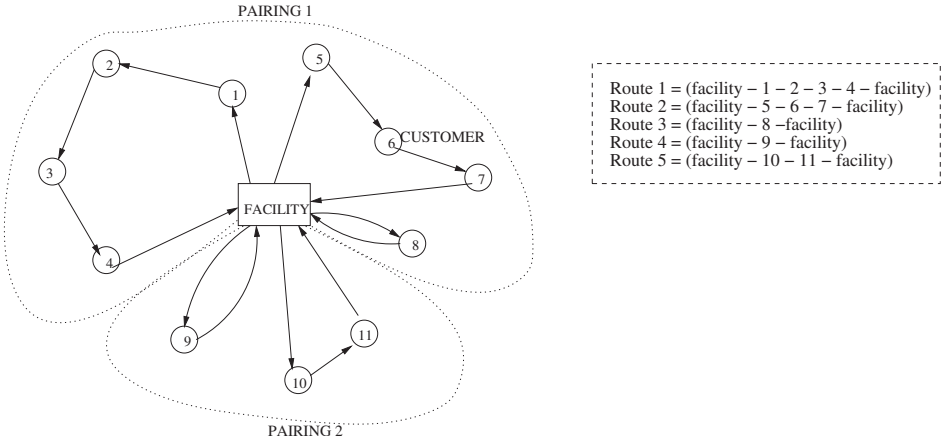


Figure 2.1: Example of pairings

We develop models for the delivery form of the LRSP and assume that there is no service time at the customers. The models, however, can be adapted easily for pickup problems and for nonzero service times. We assume that the demand information and distance matrix are known and fixed. We assume that each facility has enough vehicles to serve the allocated customers. Furthermore, we consider one class of vehicles, all having the same characteristics and working under the same conditions.

2.2 Problem Formulations

At the heart of the LRSP is a network flow structure, deriving from the routing of customer demands, that is constrained through the vehicle and facility capacities. We consider both *graph-based* and *path-based* formulations for the LRSP, as each has been widely applied for related problems. In Section 2.2.1, we present a three-index commodity flow formulation of the LRSP. In Section 2.2.2, we present a set-partitioning-based formulation of the problem.

2.2.1 Graph-Based Formulation of LRSP

For problems related to the LRP and the VRP, the literature contains two main types of graph-based models, vehicle flow and commodity flow. Recall that in both formulations, integer variables represent the number of times a vehicle traverses a given arc or edge in an underlying graph. For the LRP and the VRP, two-index vehicle flow formulations are preferred in solution algorithms based on branch-and-cut, since these formulations include a smaller number of variables and their LP relaxations yield better bounds. However, it is not possible to formulate the LRSP using variables with only two indices because of the time constraint for the vehicles. In this context, we require a three-index commodity flow model. To formulate the model, we introduce the following notation.

Sets

$$
\begin{aligned}
I &= \text{set of customer locations,} \\
J &= \text{set of candidate facility locations,} \\
N &= I \cup J = \text{set of all nodes,} \\
A &= (J \times I) \cup (I \times N) = \text{set of arcs,} \\
H_j &= \text{set of vehicles located at facility } j, \forall j \in J, \text{ such that } |H_j| = |I|, \\
H &= \bigcup_{j \in J} H_j = \text{set of all vehicles.}
\end{aligned}
$$

Parameters

C_j^F = daily equivalent fixed cost of opening facility j, $\forall j \in J$,

C^O = vehicle operating cost per unit travel time,

C^V = daily equivalent fixed cost of a vehicle (including the driver cost),

D_i = demand of customer i, $\forall i \in I$,

L_j^F = capacity of facility j, $\forall j \in J$,

L^V = capacity of a vehicle,

L^T = time limit for a vehicle and the driver, and

T_{ij} = travel time between locations i and j, $\forall (i,j) \in A$.

Decision Variables

$$x_{ikh} = \begin{cases} 1 & \text{if vehicle } h \text{ travels on arc } (i,k), \forall h \in H, (i,k) \in A, \\ 0 & \text{otherwise,} \end{cases}$$

y_{ikh} = flow on arc (i,k) carried by vehicle h, $\forall (i,k) \in A, h \in H$,

$$t_j = \begin{cases} 1 & \text{if facility } j \text{ is selected, } \forall j \in J, \\ 0 & \text{otherwise,} \end{cases}$$

$$v_h = \begin{cases} 1 & \text{if vehicle } h \text{ is used, } \forall h \in H, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Based on this notation, we develop the commodity flow model for the LRSP, which we refer to as (G-LRSP) in the rest of the thesis.

(G-LRSP)

$$\text{Min} \quad \sum_{j\in J} C_j^F t_j + C^V \sum_{h\in H} v_h + C^O \sum_{h\in H} \sum_{(i,k)\in A} T_{ik} x_{ikh} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{k\in N} x_{ikh} - \sum_{k\in N} x_{kih} = 0 \qquad \forall i \in N, h \in H, \tag{2.2}$$

$$\sum_{h\in H} \sum_{k\in N} x_{ikh} = 1 \qquad \forall i \in I, \tag{2.3}$$

$$\sum_{h\in H_j} \sum_{k\in I} y_{jkh} - L_j^F t_j \leq 0 \qquad \forall j \in J, \tag{2.4}$$

$$y_{ikh} - L^V x_{ikh} \leq 0 \qquad \forall (i,k) \in A, h \in H, \tag{2.5}$$

$$\sum_{k\in N} y_{ikh} - \sum_{k\in N} y_{kih} + D_i \sum_{k\in N} x_{ikh} = 0 \qquad \forall i \in I, h \in H, \tag{2.6}$$

$$\sum_{(i,k)\in A} T_{ik} x_{ikh} - L^T v_h \leq 0 \qquad \forall h \in H, \tag{2.7}$$

$$x_{jkh} = 0 \qquad \forall j \in J, k \in N, h \in H_t, t \in J \backslash \{j\}, \tag{2.8}$$

$$x_{ikh} \in \{0,1\} \quad \forall (i,k) \in A, h \in H, \tag{2.9}$$

$$y_{ikh} \geq 0 \qquad \forall (i,k) \in A, h \in H, \tag{2.10}$$

$$t_j \in \{0,1\} \quad \forall j \in J, \text{ and} \tag{2.11}$$

$$v_h \in \{0,1\} \quad \forall h \in H. \tag{2.12}$$

The objective function (2.1) states that the total cost, which includes the fixed cost of the selected facilities, the fixed cost of the vehicles (including the driver cost), and the operating cost of the vehicles, should be minimized. Constraints (2.2) require that a vehicle should enter and leave a node an equal number of times. Constraints (2.3) specify that exactly one vehicle must service customer node i. Constraints (2.4) ensure that the total flow to the customer nodes from each facility does not exceed its capacity. Constraints (2.5) are vehicle capacity constraints and define the relationship between the binary variable x_{ikh} and the flow variable y_{ikh}; each constraint ensures that the flow between any pair of nodes cannot exceed the capacity of a vehicle. Constraints (2.6) require conservation of flow at each customer node; these constraints must be

modified for pick-up problems.Constraints (2.5) and (2.6) together ensure that no route violates the vehicle capacity. The combined sets of constraints (2.2), (2.3), and (2.6) ensure that only valid routes that include a facility are formed, as in Nambiar et al. (1981). Constraints (2.7) limit the total time of a vehicle's schedule to the time limit. In this research, we assume that the set of vehicles in two different facilities are disjoint and that a vehicle located at facility j can only visit customer nodes and the facility j during its trip. Therefore, constraints (2.8) restrict travel on the arcs originating from a facility to vehicles located at that facility. Constraints (2.9), (2.10), (2.11), and (2.12) are the integrality and non-negativity requirements on the variables.

2.2.2 Set Partitioning-Based Formulation of LRSP

In this section, we introduce a set-partitioning-based formulation of the LRSP in which the pairing concept defined in Section 2.1 is used to formulate the LRSP. Given a set of candidate facility locations and the set of feasible pairings that can be constructed for each candidate facility, the objective of the LRSP is to select a subset of the facilities and an associated subset of pairings in such a way as to minimize the total cost subject to the following constraints:

a. each customer must be covered by exactly one pairing,

b. the total demand of the customers in the pairings assigned to a facility must not exceed the capacity of the facility, and

c. vehicles cannot be associated with a facility unless it is open.

To present the formulation, we define additional parameters and decision variables. Associated with each candidate facility $j \in J$, we let P_j be the set of all feasible pairings for facility j.

Parameters

$$a_{ip} = \begin{cases} 1 & \text{if customer node } i \text{ is in pairing } p \text{ of facility } j, \forall i \in I, p \in P_j, j \in J, \\ 0 & \text{otherwise, and} \end{cases}$$

C_p = cost (vehicle fixed cost plus operating cost) of pairing p of facility $j, \forall p \in P_j, j \in J$.

Decision Variables

$$t_j = \begin{cases} 1 & \text{if facility } j \text{ is selected, } \forall j \in J, \\ 0 & \text{otherwise,} \end{cases}$$

$$z_p = \begin{cases} 1 & \text{if pairing } p \text{ is selected for facility } j, \forall p \in P_j, j \in J, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Based on the notation, we develop the set-partitioning-based model for the LRSP which we refer to as (SPP-LRSP) in the rest of the thesis.

(SPP-LRSP)

$$\text{Min} \quad \sum_{j \in J} C_j^F t_j + \sum_{j \in J} \sum_{p \in P_j} C_p z_p \tag{2.13}$$

$$\text{s.t} \quad \sum_{j \in J} \sum_{p \in P_j} a_{ip} z_p = 1 \qquad \forall i \in I, \tag{2.14}$$

$$\sum_{p \in P_j} \sum_{i \in I} a_{ip} D_i z_p - L_j^F t_j \leq 0 \qquad \forall j \in J, \tag{2.15}$$

$$t_j \in \{0,1\} \quad \forall j \in J, \text{ and} \tag{2.16}$$

$$z_p \in \{0,1\} \quad \forall p \in P_j, j \in J. \tag{2.17}$$

The objective function (2.13) seeks to minimize the total cost including the fixed cost of the selected facilities and the cost of the selected pairings. We note that the cost C_p of a pairing includes both the vehicle fixed cost and variable costs. Constraints (2.14) guarantee that each customer

location is served by exactly one pairing. Constraints (2.15) ensure that the total demand of the selected pairings does not exceed the facility capacity. Constraints (2.16) and (2.17) are standard binary restrictions on the variables. We use notation S_{SPP} to denote the set of feasible solutions to (2.14) - (2.17).

The structure of the (SPP-LRSP) is very similar to the CFLP with single sourcing (CFLPSS). In the CFLPSS, only pairings with one customer are considered and assigned to capacitated facilities. However, in the LRSP, pairings with multiple customers must be taken into account.

2.3 Simple Valid Inequalities

In this section, we discuss how we can strengthen the set-partitioning-based formulation (SPP-LRSP) by adding simple valid inequalities a priori. First, we introduce a lower bound on the total number of facilities required in any integer feasible solution. Let R denote a lower bound on the minimum number of required facilities. Then we can add to (SPP-LRSP) a constraint to force the number of selected facilities to be at least R:

$$\sum_{j \in J} t_j \geq R \quad \forall (t, z) \in S_{SPP}. \tag{2.18}$$

For any instance, we can calculate a valid value for R from the demands of the customers and the capacities of the facilities. There are two possible cases:

Case 1. If all of the facilities have the same capacity value, $L_j^F = k \; \forall j \in J$, then

$$R = \left\lceil \frac{\sum_{i \in I} D_i}{k} \right\rceil .$$

Case 2. If at least one facility capacity differs, then a lower bound on the number of required facilities can be calculated using a greedy algorithm: compute the total demand and assign it to the facilities in non-increasing order of their capacities. Let $n = |J|$ and let $(j_1, j_2, .., j_n)$ be an

ordering of the facilities such that $L^F_{j_1} \geq L^F_{j_2} \geq ... \geq L^F_{j_n}$. Then,

$$R = \text{argmin}_{\{l=1..n\}} \left(\sum_{s=1}^{l} L^F_{j_s} \geq \sum_{i \in I} D_i \right).$$

Constraint (2.18) is not necessary in the formulation of the LRSP. However, as a simple lower bound on the number of required facilities, it may eliminate some solutions in which a fractional number of facilities is used, thereby improving the LP relaxation bound. The following proposition can be stated:

Proposition 2.1 *The Chvátal-Gomory rank (CG rank) of constraint (2.18) is 1, if* $\left(\frac{\sum_{i \in I} D_i}{k} \right)$ *is fractional in case 1 and* $\left(\frac{\sum_{i \in I} D_i}{L^F_{j_1}} \right)$ *is fractional or strictly less than* R *in case 2.*

Proof

Case 1: The facilities have identical capacity values.

To prove the proposition, consider the summation of constraints (2.15) over all facilities $j \in J$:

$$\sum_{j \in J} \sum_{p \in P_j} \sum_{i \in I} a_{ip} D_i z_p \leq k \sum_{j \in J} t_j \quad \forall (t, z) \in S_{SPP}. \tag{2.19}$$

If we then substitute constraint (2.14) in (2.19) and divide by k, we get:

$$\sum_{j \in J} t_j \geq \frac{\sum_{i \in I} D_i}{k} \quad \forall (t, z) \in S_{SPP}. \tag{2.20}$$

The t_j variables are binary, so $\sum_{j \in J} t_j$ must be integer. Therefore, with $R = \left\lceil \frac{\sum_{i \in I} D_i}{k} \right\rceil$, constraint (2.18) is valid. Furthermore, if the right hand side of constraint (2.20) is fractional, then constraint (2.18) has CG rank 1. Otherwise, its rank is 0, since it is the non-negative linear combination of the original inequalities.

Case 2: The facilities have different capacity values.

To generalize this statement to problems in which the facility capacities might be different, we

reorder the facilities as $t_{[1]}, t_{[2]}, .., t_{[J]}$ such that $L^F_{[1]} \geq L^F_{[2]} \geq .. \geq L^F_{[J]}$. Then, inequality (2.19) becomes

$$\sum_{j\in J}\sum_{p\in P_j}\sum_{i\in I} a_{ip}D_i z_p \leq \sum_{j\in J} L^F_{[j]} t_{[j]} \quad \forall (t,z) \in S_{SPP}. \tag{2.21}$$

Again, if we substitute in (2.21) using constraint (2.14) and then divide each side by $L^F_{[1]}$, we get:

$$t_{[1]} + \frac{L^F_{[2]}}{L^F_{[1]}} t_{[2]} + .. + \frac{L^F_{[J]}}{L^F_{[1]}} t_{[J]} \geq \frac{\sum_{i\in I} D_i}{L^F_{[1]}} \quad \forall (t,z) \in S_{SPP}. \tag{2.22}$$

If we let

$$R_2 = \left\lceil \frac{\sum_{i\in I} D_i}{L^F_{[1]}} \right\rceil,$$

then, by rounding up the coefficients of inequality (2.22), we get:

$$\sum_{j\in J} t_j \geq R_2 \quad \forall (t,z) \in S_{SPP}. \tag{2.23}$$

The value of R_2 is calculated based on the assumption that all facilities have a capacity equal to the largest of the facility capacity values. However, in the procedure for calculating R for problems that have different capacity values (case 2), it is considered that the second facility might not have a capacity as large as the first facility, the third facility might not have a capacity as large as the second facility and so on. Therefore, $R \geq R_2$ and constraint (2.18) dominates constraint (2.23). If $\left(\frac{\sum_{i\in I} D_i}{L^F_{[1]}}\right)$ is fractional or $R_2 < R$, then the CG rank of constraint (2.18) is 1. □

Due to the similarity in structure between the (SPP-LRSP) and the CFLP, we might consider adding the set of constraints:

$$z_p \leq t_j \quad \forall j \in J, p \in P_j, \quad \forall (t,z) \in S_{SPP}. \tag{2.24}$$

In the CFLP, constraints similar to inequality (2.24) induce facets of the CFL polytope if the demands of the customers in the constraints are less than the capacity of the facilities (Leung and Magnanti (1989)). In addition, Daskin (1995) remarks that constraints in the CFLP similar to (2.24) considerably improve the LP bound.

Constraints (2.24) ensure that a pairing of a facility can be selected only if that facility is selected. Although constraints (2.24) are implied by constraints (2.15) when integrality is enforced, their presence may improve the LP relaxation of the model as in the case of CFLP. However, since there is potentially an exponential number of such constraints, we would like to find a smaller set of constraints to add.

Proposition 2.2 *Constraints*

$$\sum_{p \in P_j} a_{ip} z_p \leq t_j \quad \forall j \in J, i \in I, \quad \forall (t,z) \in S_{SPP}, \tag{2.25}$$

have a CG rank of 1 and dominate constraints (2.24).

Proof In any feasible solution, each customer node $i \in I$ is assigned to exactly one facility, so $\sum_{p \in P_j} a_{ip} z_p = 1$ for exactly one facility j and 0 for all others. For any $j \in J$ such that $\sum_{p \in P_j} a_{ip} z_p = 1$, t_j must be 1, since pairings can only be for open facilities (stated by constraints 2.15 and integrality restrictions).

Also, when the coefficients of the variables in inequality (2.25) and constraints (2.24) are compared, it is easy to see that inequality (2.25) dominates constraint (2.24). For example, if we take $z_{p'} \leq t_{j'}$ from constraint set (2.24) for $j = j'$ and $p = p'$, we can compare it with a constraint in set (2.25) for which $j = j'$ and $i = i'$ such that $a_{i'p'} = 1$, i.e., $z_{p'} + \sum_{p \in P_{j'} \setminus \{p'\}} a_{i'p} z_p \leq t_{j'}$. Because of the terms involving variables other than $z_{p'}$ on the left side of the constraint, $\sum_{p \in P_{j'}} a_{i'p} z_p \leq t_{j'}$ dominates $z_{p'} \leq t_{j'}$.

Now we compute the rank of the inequality $\sum_{p \in P_{j'}} a_{i'p} z_p \leq t_{j'}$ for some $i' \in I$ and $j' \in J$. Let $D(k) = \sum_{i \in I} a_{ik} D_i$ be the total demand of a pairing $k \in P_{j'}$. Using constraints (2.14) and

(2.15), we can derive the inequalities:

$$\sum_{\{p\in P_{j'}|a_{i'p}=1\}} z_p + \sum_{\{j\in J\setminus\{j'\}\}} \sum_{\{p\in P_j|a_{i'p}=1\}} z_p \leq 1 \quad \forall (t,z) \in S_{SPP}, \tag{2.26}$$

$$\sum_{\{p\in P_{j'}|a_{i'p}=1\}} D(p)z_p + \sum_{\{p\in P_{j'}|a_{i'p}=0\}} D(p)z_p - L^F_{j'}t_{j'} \leq 0 \quad \forall (t,z) \in S_{SPP}. \tag{2.27}$$

Let $D_{min} = \min_{\{p\in P_{j'}|a_{i'p}=1\}}\{D(p)\}$. Multiplying inequality (2.26) by $(L^F_{j'} - D_{min})$, adding the resulting inequality to the inequality (2.27), dividing the obtained inequality by $L^F_{j'}$ and rounding, we obtain:

$$\begin{aligned}\sum_{\{p\in P_{j'}|a_{i'p}=1\}} \left\lfloor \frac{L^F_{j'} - D_{min} + D(p)}{L^F_{j'}} \right\rfloor z_p \; &+ \sum_{\{j\in J/\{j'\}\}} \sum_{\{p\in P_j|a_{i'p}=1\}} \left\lfloor \frac{L^F_{j'} - D_{min}}{L^F_{j'}} \right\rfloor z_p \\ &+ \sum_{\{p\in P_{j'}|a_{i'p}=0\}} \left\lfloor \frac{D(p)}{L^F_{j'}} \right\rfloor z_p - t_{j'} \\ &\leq \left\lfloor \frac{L^F_{j'} - D_{min}}{L^F_{j'}} \right\rfloor \quad \forall (t,z) \in S_{SPP}. \end{aligned} \tag{2.28}$$

Note that we assume that $D(p) \leq L^F_{j'} \;\; \forall p \in P_{j'}$. Because if the total demand of the pairing is greater than the capacity of the facility, this pairing cannot be chosen in a feasible solution for this facility and thus we can set $z_p = 0$ and remove the variable from the constraint.

In inequality (2.28), the coefficients in the first summation are in the range $[1, 2)$, the coefficients in the second summation are in $[0, 1)$, and the coefficients in the third summation are in the range $[0, 1)$ if $D(p) < L^F_{j'}$ for all $p \in P_j$. Furthermore, the right hand side is in the range $[0, 1)$. Thus, when we round down the coefficients, we get

$$\sum_{p\in P_{j'}} a_{i'p} z_p \leq t_{j'} \quad \forall (t,z) \in S_{SPP}.$$

Since we have obtained inequality (2.25) by using the original constraints and the CG inequality procedure, the CG rank of the inequality is 1. Also, this inequality is not dominated by any linear

combination of the original constraints, thus its CG rank cannot be 0. If $D(p) = L^F_{j'}$ for some $p \in P_{j'}$ in the third summation, when we round down, the coefficient for this variable in (2.28) is 1. Thus, we can add the variables z_p for all $p \in P_{j'}$ and satisfying $D(p) = L^F_{j'}$ and $a_{i'p} = 0$ to inequality (2.25) to get a stronger constraint. □

Experiments evaluating the effectiveness of inequalities (2.18), (2.24) and (2.25) are presented in Section 5.3.

2.4 Relationship Between Graph- and Set Partitioning-based Formulations

In this section, we explore the relationship between the formulations of the LRSP presented earlier in the chapter. We prove that (SPP-LRSP) can be obtained by applying Dantzig-Wolfe decomposition (DWD) to the graph-based formulation of the LRSP (G-LRSP). In Section 1.2.9, the DWD methodology is explained for a general MILP model. Here, we provide the details of the procedure for the LRSP.

2.4.1 Dantzig-Wolfe Decomposition of the Graph-based Formulation

To reformulate (G-LRSP) using DWD, we must decompose the formulation into two subsystems: master problem and subproblem. The *master problem* is defined by constraints (2.3) and (2.4), while the *subproblem* is defined by constraints (2.2), and (2.5) - (2.12). Recall that the vehicle index set, H is the union of the vehicle index sets for each facility, i.e., $H = \cup_{j \in J} H_j$. Thus, all constraints of the subproblem are indexed by the set J and it is therefore immediate that the subproblem decomposes by facility.

For each candidate facility $j \in J$, the set of integer solutions of the decomposed subproblem

can be represented by the set of vectors in the set

$$\{\ (x,t,v)^j \in \mathbb{Z}^{(|A|\times|H_j|)+\{0,1\}+(|H_j|)}, y^j \in \mathbb{R}^{(|A|\times|H_j|)} \mid\ (x,y,t,v)^j \text{ satisfies}$$
$$(2.2)_j, (2.5)_j, (2.6)_j, (2.7)_j, (2.8)_j, (2.9)_j, (2.10)_j, (2.11)_j, (2.12)_j\}, \tag{2.29}$$

which is described only by those constraints associated with facility j and specifies values only for those variables associated with vehicles assigned to facility j. Hence, the index j for the constraints indicates the set of constraints associated with facility j. Set (2.29) is bounded for all facilities. We let E be a set indexing the solutions in all of the above sets, with E_j the indices for vectors associated only with facility j, so that $E = \cup_{j\in J} E_j$. For a facility j and an index $q \in E_j$, the corresponding member $(\mathrm{x}^q, \mathrm{y}^q, \mathrm{t}^q, \mathrm{v}^q)$ of the above set is then a vector with the following interpretation:

$$\mathrm{x}^q_{ikh} = \begin{cases} 1 & \text{if vehicle } h \text{ travels on arc } (i,k) \text{ in solution } q, \forall (i,k) \in A, \text{ and } h \in H_j, \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathrm{y}^q_{ikh} = \text{flow on arc } (i,k) \text{ carried by vehicle } h \text{ in solution } q, \forall (i,k) \in A, \text{ and } h \in H_j,$$

$$\mathrm{t}^q = \begin{cases} 1 & \text{if facility } j \text{ is selected in solution } q, \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathrm{v}^q_h = \begin{cases} 1 & \text{if vehicle } h \text{ is used in solution } q, \forall h \in H_j, \\ 0 & \text{otherwise.} \end{cases}$$

Note that the variable t indicating whether the facility is open does not appear in any of the linear constraints of the subproblem and can hence be set to either 0 or 1 without affecting feasibility.

Following the DWD methodology, we can reformulate (G-LRSP) by replacing the subproblem constraints with a requirement that solutions are convex combinations of the feasible points of the subproblem. Because the subproblem decomposes as described above, solutions to the original problem can be seen as vectors obtained by "recomposing" convex combinations of the members

of E_j for each $j \in J$. In other words, we express solutions (x, y, t, v) to formulation (G-LRSP) as

$$x_{ikh} = \sum_{q \in E} \mathrm{x}^q_{ikh} \theta_q \quad \forall (i,k) \in A, h \in H, \tag{2.30}$$

$$y_{ikh} = \sum_{q \in E} \mathrm{y}^q_{ikh} \theta_q \quad \forall (i,k) \in A, h \in H, \tag{2.31}$$

$$t_j = \sum_{q \in E_j} \mathrm{t}^q \theta_q \quad \forall j \in J, \tag{2.32}$$

$$v_h = \sum_{q \in E} v^q_h \theta_q \quad \forall h \in H, \tag{2.33}$$

$$\sum_{q \in E_j} \theta_q = 1 \quad \forall j \in J, \tag{2.34}$$

$$\theta_q \geq 0 \quad \forall q \in E, \tag{2.35}$$

where

$$\theta_q = \begin{cases} 1 & \text{if solution vector } (\mathrm{x}^q, \mathrm{y}^q, \mathrm{t}^q, \mathrm{v}^q) \text{ for subproblem is used, } \forall q \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Then, we rewrite the objective function (2.1), and the master problem constraints (2.3) and (2.4) based on the relationship between solutions (x, y, t, v) and the set of solutions of the subproblem defined by (2.30) - (2.35). The following linear master problem can be obtained.

($\mathbf{G_{MLP0}}$)

$$\text{Min} \quad \sum_{q \in E} (C^F_q \mathrm{t}^q + C^V \sum_{h \in H} \mathrm{v}^q_h + C^O \sum_{i \in N} \sum_{k \in N} \sum_{h \in H} T_{ik} \mathrm{x}^q_{ikh}) \theta_q$$

$$\text{s.t.} \quad \sum_{q \in E} (\sum_{h \in H} \sum_{k \in N} \mathrm{x}^q_{kih}) \theta_q = 1 \quad \forall i \in I, \tag{2.36}$$

$$-\sum_{q \in E_j} (\sum_{h \in H_j} \sum_{i \in I} \mathrm{y}^q_{jih}) \theta_q + L^F_j \sum_{q \in E_j} \mathrm{t}^q \theta_q \geq 0 \quad \forall j \in J, \tag{2.37}$$

$$\sum_{q \in E_j} \theta_q = 1 \quad \forall j \in J, \tag{2.38}$$

$$\theta_q \geq 0 \quad \forall q \in E. \tag{2.39}$$

We can simplify the formulation (G_{MLP0}) using some substitutions. First, we rewrite constraint (2.37) by deriving a relationship between x^q and y^q based on (G-LRSP). We sum flow conservation constraints (2.6) over all $i \in I$ and observe that variables representing the flow between two customer nodes (y_{ikh} $\forall i, k \in I$ and $h \in H$) cancel each other out. In addition, by using the information that in any feasible solution, the vehicle returns to the facility (variable y_{ijh} for some facility j and vehicle h and for all $i \in I$) with zero load because of the characteristics of equations (2.6), we obtain the equation

$$\sum_{i \in I} y_{jih} = \sum_{i \in I} D_i \sum_{k \in N} x_{ikh} \quad \forall j \in J, h \in H_j, \tag{2.40}$$

which can be used to derive the equation

$$\sum_{q \in E_j} \sum_{i \in I} y^q_{jih} \theta_q = \sum_{q \in E_j} \sum_{i \in I} D_i \sum_{k \in N} x^q_{ikh} \theta_q \quad \forall j \in J. \tag{2.41}$$

Substituting equality (2.41) and defining additional notation, we simplify (G_{MLP0}) further:

$$\textbf{(G}_{\textbf{MLP}}\textbf{)} \qquad \text{Min} \quad \sum_{q \in E} \tilde{C}_q \theta_q$$

$$\text{s.t.} \quad \sum_{q \in E} b_{iq} \theta_q = 1 \quad \forall i \in I, \tag{2.42}$$

$$-\sum_{q \in E_j} \sum_{i \in I} b_{iq} D_i \theta_q + L^F_j \sum_{q \in E_j} t^q \theta_q \geq 0 \quad \forall j \in J, \tag{2.43}$$

$$\sum_{q \in E_j} \theta_q = 1 \quad \forall j \in J, \tag{2.44}$$

$$\theta_q \geq 0 \quad \forall q \in E, \tag{2.45}$$

where

$$
\begin{aligned}
b_{iq} &= \sum_{h\in H}\sum_{k\in N} \mathrm{x}^q_{kih}, && \forall i \in I, q \in E,\\
\tilde{C}_q &= C^F_q \mathrm{t}^q + C^V \sum_{h\in H} \mathrm{v}^q_h + C^O \sum_{i\in N}\sum_{k\in N}\sum_{h\in H} T_{ik}\mathrm{x}^q_{ikh}, && \forall q \in E,
\end{aligned}
$$

where $C^F_q = C^F_j$ when $q \in E_j$ for $j \in J$. Here, b_{iq} can be interpreted as the number of times customer i is visited in solution q, and $\tilde{C}_q$ is the cost of solution q (including facility fixed cost and vehicle fixed costs) for all $q \in E$.

Any feasible solution $\bar{\theta}$ to ($\mathrm{G_{MLP}}$) can be transformed into vector $(\bar{x}, \bar{y}, \bar{t}, \bar{v})$ using (2.30) - (2.33). However, this solution may not be integer feasible for (G-LRSP). To be able to obtain a true reformulation of (G-LRSP) using DWD, we also need to satisfy integrality restrictions for (x, t, v). We can do that by adding integrality restriction to θ in ($\mathrm{G_{MLP}}$). We call the following problem the integer master problem and refer to it as ($\mathrm{G_M}$).

$$
\begin{aligned}
& \text{Min} && \sum_{q\in E} \tilde{C}_q\theta_q && && \\
\textbf{(G}_{\mathbf{M}}\textbf{)} \qquad & \text{s.t.} && \sum_{q\in E} b_{iq}\theta_q = 1 && \forall i \in I,\\
& && -\sum_{q\in E_j}\sum_{i\in I} b_{iq}D_i\theta_q + L^F_j \sum_{q\in E_j} \mathrm{t}^q\theta_q \geq 0 && \forall j \in J,\\
& && \sum_{q\in E_j} \theta_q = 1 && \forall j \in J,\\
& && \theta_q \in \{0,1\} && \forall q \in E.
\end{aligned}
$$

To solve ($\mathrm{G_M}$), we can apply a branch-and-price algorithm (see Section 1.2.10). The LP relaxation of the ($\mathrm{G_M}$), which is ($\mathrm{G_{MLP}}$), is solved using a column generation algorithm (described in Section 1.2.8) to obtain a lower bound. Let $(\pi, \mu, \alpha)^\top$ be a dual vector such that π is the dual variable associated with constraint (2.42), μ is the dual variable associated with (2.43), and α is the dual variable associated with (2.44). The reduced cost of a column with index $q \in E_j$ for some

$j \in J$ corresponding to variable θ_q is

$$\hat{C}_q = \tilde{C}_q - \sum_{i \in I} b_{iq}\pi_i + \left(\sum_{i \in I} b_{iq} D_i - L_j^F \mathrm{t}^q \right) \mu_j - \alpha_j. \tag{2.46}$$

Using the definitions of $\tilde{C}$, we can rewrite (2.46):

$$\begin{aligned} \hat{C}_q \;\; = \;\; & (C_j^F - L_j^F \mu_j)\mathrm{t}^q + C^V \sum_{h \in H_j} \mathrm{v}_h^q - \\ & \sum_{h \in H_j} \sum_{k \in N} \Big(\sum_{i \in I} (C^O T_{ki} - \pi_i + \mu_j D_i)\mathrm{x}_{kih}^q + C^O T_{kj} \mathrm{x}_{kjh}^q \Big) - \alpha_j. \end{aligned} \tag{2.47}$$

The column generation algorithm tries to determine the column of ($\mathrm{G_M}$) with the most negative reduced cost, as defined by (2.47), where the candidate columns are the set for facility j defined in (2.29). This problem is called the subproblem for facility $j \in J$ (referred to as (SP_j)), and can be formulated explicitly as follows:

($\mathbf{SP}_j$)

$$\text{Min} \quad \hat{f}_j t_j + C^V \sum_{h \in H_j} v_h - \sum_{h \in H_j} \sum_{k \in M} \sum_{i \in M} \hat{c}_{ki} x_{kjh} - \alpha_j$$

$$\text{s.t.} \quad \sum_{k \in M} x_{ikh} - \sum_{k \in M} x_{kih} = 0 \qquad \forall i \in M, h \in H_j, \tag{2.48}$$

$$y_{ikh} - L^V x_{ikh} \leq 0 \qquad \forall i \in M, k \in M, h \in H_j, \tag{2.49}$$

$$\sum_{k \in M} y_{ikh} - \sum_{k \in M} y_{kih} + D_i \sum_{k \in M} x_{ikh} = 0 \qquad \forall i \in I, h \in H_j, \tag{2.50}$$

$$\sum_{i \in M} \sum_{k \in M} T_{ik} x_{ikh} - L^T v_h \leq 0 \qquad \forall h \in H_j, \tag{2.51}$$

$$x_{ikh} \in \{0,1\} \qquad \forall i \in M, k \in M, h \in H_j, \tag{2.52}$$

$$y_{ikh} \geq 0 \qquad \forall i \in M, k \in M, h \in H_j, \tag{2.53}$$

$$t_j \in \{0,1\}, \quad \text{and} \tag{2.54}$$

$$v_h \in \{0,1\} \qquad \forall h \in H_j, \tag{2.55}$$

where $M = I \cup \{j\}$ and

$$\begin{aligned} \hat{f}_j &= C_j^F - L_j^F \mu_j, \\ \hat{c}_{ki} &= \begin{cases} C^O T_{ki} - \pi_i + \mu_j D_i & \text{if } k \in M, \text{ and } i \in I, \\ C^O T_{kj} & \text{if } k \in I. \end{cases} \end{aligned}$$

The objective of each of the resulting single-facility subproblems (SP_j) is to generate sets of paths with least-cost assigned to *all vehicles* located at the facility, though without the constraint that customers appear exactly once. Because of this condition, we avoid calling these paths "routes" for now.

Now we prove that solving (G_M) is equivalent to solving (SPP-LRSP). The similar forms of (SPP-LRSP) and (G_M) should now be evident, but to rigorously show their equivalence, we need to dissect the relationship between set E_j (used to define (G_M)) and P_j (used to define (SPP-LRSP)) for a given facility $j \in J$.

A member of set E_j, a feasible solution to (SP_j), consists of a *collection* of sets of paths (starting at facility j and ending at facility j) assigned to all vehicles located at facility j. It may be possible that some customers are visited multiple times by multiple vehicles. A member of set P_j, on the other hand, is a single set of routes (pairing) that can be assigned to any vehicle at facility j. A customer can only appear at most once in a pairing assigned to a vehicle.

Certain members of set E_j for $j \in J$ can be constructed by associating at most $|H_j|$ members of set P_j and some number of empty pairings (zero vectors representing vehicles that are not used) to the vehicles assigned to facility j. The members of E_j that include sets of paths assigned to a single vehicle and visiting some customers multiple times cannot be constructed using members of P_j.

By utilizing the integrality requirements from the original problem and carefully eliminating the indices of symmetric solutions from E_j, we get a much smaller set that we will show is in

one-to-one correspondence with collections of members of P_j of cardinality at most $|H_j|$. We proceed as follows:

1. Let $q \in E$ be a member in which a vehicle visits at least one customer multiple times. Thus, $\mathrm{x}^q_{ik_1h_1} = 1$ and $\mathrm{x}^q_{ik_2h_1} = 1$ for some customer $i \in I$, vehicle $h_1 \in H$ and nodes $k_1, k_2 \in M$ such that $k_1 \neq k_2$. This means that $\sum_{k\in M} \mathrm{x}^q_{ikh_1} > 1$ and therefore $b_{iq} > 1$. Because of constraint (2.42) and the binary restriction for θ_q, a solution to ($\mathrm{G_M}$) for which the corresponding solution in (G-LRSP) is feasible, θ_q must be 0. Therefore, we can remove such members from E. In order not to generate these members, we can add the following constraint to (SP_j),

$$\sum_{k\in M} x_{ikh} \leq 1 \quad \forall i \in I, h \in H_j. \tag{2.56}$$

   When we add constraint (2.56) to (SP_j), the feasible solutions are restricted to sets of routes each of which assigned to a vehicle and includes any customer at most once. Therefore, based on the definition of a pairing in Section 2.2.2, we can say that the feasible solutions to augmented (SP_j) are sets of pairings.

2. As the vehicles associated with a given facility j are identical, a set of pairings can be assigned to vehicles in any arbitrary order. Hence, we obtain different members of E_j that are all equivalent from the standpoint of both feasibility and cost. To eliminate superfluous equivalent members of E_j, we divide the members of set E_j into equivalence classes, where two members of E_j are considered equivalent if the set of customers assigned to the facility and the partition of that set of customers defined by the pairings (set of routes assigned to a vehicle) are identical. It is clear that any two members of E_j that are equivalent by this definition will have exactly the same impact on both cost and feasibility.

   In each equivalence class of E_j there can be up to $|H_j|!$ members depending on the number of vehicles serving to non zero pairings in the solutions. For example, let q_1 and q_2 be two

members of E_j and be in the same equivalence class. Then there exists a pair of vehicles h_1 and h_2 such that $\mathrm{x}^{q_1}_{ikh_1} = \mathrm{x}^{q_2}_{ikh_2}$, $\mathrm{x}^{q_1}_{ikh_2} = \mathrm{x}^{q_2}_{ikh_1}$ $\forall i, k \in M$ and at least one of h_1 and h_2 is serving a non zero pairing.

We form equivalence classes from which all but one member may safely be eliminated from E_j.

3. Index q can be removed from E_j if there exist vehicles $h_1, h_2 \in H_j$ such that $\mathrm{x}^q_{ikh_1} = \mathrm{x}^q_{ikh_2}$ $\forall i, k \in M$, where $\mathrm{x}^q_{ikh_1} > 0$ for some $i, k \in M$ (i.e., this is not an empty route). In this case, vehicles h_1 and h_2 define exactly the same sets of routes, which means that $b_{iq} > 1$ for some $i \in I$. Because of constraint (2.42), such a solution must have $\hat{\theta}_q = 0$.

4. From (2.32) and (2.43), we can conclude that if $\hat{\theta}_q = 1$ and we have $\mathrm{x}^q_{ikh} > 0$ for some $i, k \in M$ and $h \in H$, then t^q must be 1. Hence, we can eliminate any solution for which $\mathrm{t}^q = 0$ that does not correspond to a zero solution (i.e., closed facility). All of this allows us to rewrite (2.43) in the form

$$-\sum_{q \in E_j} \sum_{i \in I} b_{iq} D_i \theta_q + L^F_j \sum_{q \in E_j} \theta_q \geq 0 \;\; \forall j \in J. \tag{2.57}$$

5. A member $q \in E_j$ can be removed if there exists a vehicle $h \in H_j$ such that $\mathrm{v}^q_h = 1$ and $\mathrm{x}^q_{ikh} = 0$ for all $i, k \in M$ (meaning that zero vector is assigned to vehicle h even though we pay the fixed cost of that vehicle). In an optimal solution to ($\mathrm{G_M}$), $\theta_{q'} \geq \theta_q = 0$ where $q, q' \in E_j$, $\mathrm{x}^q = \mathrm{x}^{q'}$, $\mathrm{y}^q = \mathrm{y}^{q'}$, $\mathrm{t}^q = \mathrm{t}^{q'}$, and $\mathrm{v}^q = \mathrm{v}^{q'}$ except that $\mathrm{v}^{q'}_h = 0$ since $\tilde{C}_q \geq \tilde{C}_{q'}$.

6. Finally, we can remove index q corresponding to the zero column (i.e., $\mathrm{x}^q = 0$, $\mathrm{y}^q = 0$, $\mathrm{v}^q = 0$, and $\mathrm{t}^q = 0$) from E_j. Thus, we can update constraint (2.44):

$$\sum_{q \in E_j} \theta_q \leq 1 \quad \forall j \in J. \tag{2.58}$$

If we restrict set E_j according to the rules described above and call the restricted set $\bar{E}_j$ for

each $j \in J$, then we can finally conclude the following.

Proposition 2.3 *There is a one-to-one correspondence between nonempty subsets of P_j with cardinality less than $|H_j|$ and members of $\bar{E}_j$.*

Proof The proof follows easily from the definitions of sets P_j and $\bar{E}_j$, and the restriction rules. □

By replacing set E_j with $\bar{E}_j$ for all $j \in J$ in ($\mathrm{G_M}$), as well as replacing (2.43) with (2.57) and (2.44) with (2.58) we obtain a new (equivalent) formulation ($\mathrm{G_M}'$). Then, we finally have the equivalence of (SPP-LRSP) and ($\mathrm{G_M}$) as follows.

Proposition 2.4 *There is a one-to-one correspondence between solutions to (SPP-LRSP) and solutions to (G_M') such that corresponding solutions also have the same objective function value. Thus, (SPP-LRSP), (G_M'), and (G_M) are all equivalent.*

Proof First, we construct a one-to-one mapping to establish the correspondence between solution sets of (SPP-LRSP) and ($\mathrm{G_M}'$). We define mapping $\Psi : (t, z) \rightarrow \theta$. Let $(\bar{t}, \bar{z})$ be a feasible solution to (SPP-LRSP) where $\bar{t}_j = 1$ if facility j is open, $\bar{z}_p = 1$ if pairing p is selected and assigned to a vehicle for all $p \in P_j$ and $j \in J$, then $\bar{\theta} = \Psi(\bar{t}, \bar{z})$ be the corresponding solution in ($\mathrm{G_M}'$) where $\bar{\theta}_q = 1$ if solution q is selected for all $q \in \bar{E}_j$, $j \in J$.

Since we can decompose variables in both formulations by facility, we can decompose mapping Ψ by facility and define mapping Ψ_j indexed by facility. $\Psi = (\Psi_j)_{j \in J}$ and $\bar{\theta}_{\bar{E}_j} = \Psi_j(\bar{t}_j, \bar{z}_{P_j})$ where notation x_S represents a vector composed of the components of vector x indexed with set S. Let $\bar{J} = \{j \in J | \bar{t}_j = 1\}$, $\bar{P}_j = \{p \in P_j | \bar{z}_p = 1\}$ $\forall j \in \bar{J}$ and $\bar{P}_j = \emptyset$ $\forall j \in J \backslash \bar{J}$ (because of constraint (2.15)). For some facility $j \in J$, we have three types of solutions, which changes the definition of $\bar{\theta}_{\bar{E}_j} = \Psi_j(\bar{t}_j, \bar{z}_{P_j})$.

i. Zero solution, $\bar{t}_j = 0$ and $\bar{P}_j = \emptyset$, thus $\bar{z}_p = 0$ for all $p \in P_j$. Then, $\Psi_j(\bar{t}, \bar{z}_{P_j}) = \bar{\theta}_{\bar{E}_j} = 0$, i.e., $\bar{\theta}_q = 0$ for all $q \in \bar{E}_j$ (since we removed zero solution from $\bar{E}_j$ in reduction rule 6).

ii. Nonzero but empty solution, $\bar{t}_j = 1$ and $\bar{P}_j = \emptyset$. Then, $\bar{\theta}_i = 1$ such that $\mathrm{x}^i = 0$, $\mathrm{y}^i = 0$, $\mathrm{v}^i = 0$ and $\mathrm{t}^i = \bar{t}_j = 1$, and $\bar{\theta}_q = 0$ for all $q \in \bar{E}_j$ and $q \neq i$.

iii. $\bar{t}_j = 1$ and $\bar{P}_j \neq \emptyset$. For each facility we can choose at most $|I| = |H_j|$ pairings, i.e., $\sum_{p \in P_j} \bar{z}_p = |\bar{P}_j| \leq |H_j|$. Based on Proposition 2.3, there is one member $k \in \bar{E}_j$ (in which $|\bar{P}_j|$ vehicles assigned one pairing from set $\bar{P}_j$, $|H_j| - |\bar{P}_j|$ vehicles assigned zero solution, and $\mathrm{t}_j^k = 1$) corresponds to subset $\bar{P}_j$. Thus, the corresponding solution in ($\mathrm{G_M}'$) is $\bar{\theta}_k = 1$ and $\bar{\theta}_q = 0$ for all $q \in \bar{E}_j$ and $q \neq k$. We use notation $k \rightleftharpoons \bar{P}_j$ to define this one-to-one correspondence between member k and pairing set $\bar{P}_j$.

It follows that Ψ_j is one-to-one from construction, therefore Ψ is also one-to-one. It is clear that we can easily construct the one-to-one reverse mapping, $(\bar{t}_j, \bar{z}_{P_j}) = \Psi_j^{-1}(\bar{\theta}_{\bar{E}_j})$ in a similar way.

From the construction of Ψ_j for each solution type, we can write the following relations between two formulations' variables and parameters.

$$t_j = \sum_{q \in \bar{E}_j} \theta_q \qquad \forall j \in J, \tag{2.59}$$

$$b_{iq} = \sum_{p \in S : q \rightleftharpoons S} a_{ip} \qquad \forall i \in I, \forall q \in \bar{E}_j, \tag{2.60}$$

$$\tilde{C}_q = C_j^F + \sum_{p \in S : q \rightleftharpoons S} C_p \quad \forall q \in \bar{E}_j, \forall j \in J. \tag{2.61}$$

If $(\bar{t}, \bar{z})$ is feasible, then $\bar{\theta} = \Psi(\bar{t}, \bar{z})$ is also feasible. This can be seen from the similarity between the structures of (SPP-LRSP) and ($\mathrm{G_M}'$) and from (2.59) and (2.60). Finally, the equivalence of objective function values follows from (2.61).

□

Proposition 2.4 proves that applying DWD to (G-LRSP) yields (SPP-LRSP).

2.4.2 Comparing Alternative Formulations

(G-LRSP) is a three-index commodity flow formulation which can require up to O($|I|^4$) binary variables and O($|I|^4$) constraints based on the assumption that the number of candidate facilities is at most the number of customers. In addition, because of the facility and vehicle capacity restrictions, we do not know a priori the number of vehicles required at each facility, a naive upper bound is $|I|$.

On the other hand, (SPP-LRSP) includes O($|I|$) number of constraints while the number of variables depends on the size of the pairing sets (i.e., $|\bigcup_{j\in J} P_j|$), which can be exponentially large. However, most of these variables will have value zero in an optimal solution. Therefore, intelligent enumeration techniques which may use some logical conditions, information from good solutions found, and information from experience can be developed to avoid generating all possible columns.

Observe that the three-index formulation (G-LRSP) exhibits a high degree of symmetry under the assumption that the vehicle fleet assigned to each facility is homogeneous. This is due to the fact that the assignment of pairings to a specific vehicle is essentially arbitrary, i.e., the cost of a given solution to (G-LRSP) is invariant under permutation of the indices assigned to specific vehicles. In addition, from the definition of the variables in formulation (G-LRSP), symmetric distance matrix results in solutions with same cost but including routes constructed in different directions.

(SPP-LRSP) formulation eliminates the symmetry that arises in (G-LRSP) because of the identical vehicles. In (SPP-LRSP), we enumerate one set of feasible pairings that can be assigned to any vehicle located at a facility instead of duplicating this set for each vehicle located at a facility. Furthermore, while generating the pairings using a subproblem, depending on the solution procedure, we can use undirected links between nodes in case of symmetric distance matrix. Therefore, we can remove direction in the routes.

The lower bound obtained by solving the LP relaxation of (SPP-LRSP) is stronger than the

bound obtained by solving the LP relaxation of (G-LRSP). This can be demonstrated both theoretically and empirically. Let z_{LP}^{G} be the LP relaxation of formulation (G-LRSP) and z_{LP}^{SPP} be the LP relaxation of formulation SPP-LRSP. In Proposition 2.4 we proved that ($G_M{}'$) obtained by applying Dantzig-Wolfe Decomposition to (G-LRSP) is equivalent to (SPP-LRSP). Therefore using the DWD methodology we can theoretically compare the LP relaxation bounds of (G-LRSP) and (SPP-LRSP). Let z_{LP}^{M} be the LP relaxation of the formulation ($G_M{}'$). Then, we have the following result.

Proposition 2.5 $z_{LP}^{G} \leq z_{LP}^{M} = z_{LP}^{SPP}$.

Proof z_{LP}^{G} is obtained by optimizing the objective function (2.1) on the polyhedron defined by constraints (2.2) - (2.8) ((G-LRSP) without the integrality constraints).

On the other hand, z_{LP}^{M} is obtained by optimizing the objective function (2.1) on the polyhedron defined by the intersection of constraints (2.3) and (2.4) with the convex hull of of the solutions that are constructed with composing the points in the set (2.29) for all facilities, which is the convex hull defined by (2.2), and (2.5) - (2.12).

The proof follows from the fact that the subproblems to generate the set (2.29) for all facilities do not have the integrality property. The polyhedron defined by the constraints (2.2) and (2.5) - (2.8) is greater or equal to the convex hull of the same set of constraints. □

The inequality in Proposition 2.5 can also be demonstrated empirically. In Section 5.2, we present computational results that compare z_{LP}^{G} and z_{LP}^{SPP}.

Finally, in terms of inequalities valid for the model, we can see additional differences between working with formulation (G-LRSP) and formulation (SPP-LRSP). In (G-LRSP), the variables are indexed with vehicle h, which enables us to define valid inequalities associated with either a single vehicle (i.e., only the variables associated with vehicle h has nonzero coefficient) or a set of vehicles. Let $u_h = (x_{ikh}, y_{ikh}, v_h)_{(i,k)\in A}$ represent the variable vector that is associated with vehicle $h \in H$. We may derive cut (α, β) either in the form of $\alpha u_{\bar{h}} \geq \beta$ for some vehicle $\bar{h} \in H$, or in the form of $\sum_{h\in K} \alpha_h u_h \geq \beta$ for some set K such that $K \subseteq H$. In addition, we

can multiply a cut generated for a single vehicle h for all vehicles in set H. A cut (α, β) which is derived for vehicle $\bar{h}$ and is in the form of $\alpha u_{\bar{h}} \geq \beta$, can also be written for other vehicles, e.g., $\alpha u_h \geq \beta$ for all $h \in H$. This can prevent jumps between the symmetrical solutions because of the homogeneous vehicle fleet. Let $\bar{u}_1$ and $\bar{u}_2$ be the current LP relaxation solution vectors corresponding to vehicle 1 and 2 which are in index set H_j for some facility $j \in J$, respectively. When we only add $\alpha u_{\bar{h}} \geq \beta$ to the current LP relaxation and resolve, we may obtain an LP relaxation solution such that $\bar{u}'_1 = \bar{u}_2$ and $\bar{u}'_2 = \bar{u}_1$ where $\bar{u}'_1$ and $\bar{u}'_2$ are the most recent solution vectors corresponding to vehicle 1 and 2, respectively.

In formulation (SPP-LRSP), a pairing variable for a facility is associated with any vehicle located at the facility, thus we can not differentiate the vehicles. We can only use cutting planes associated with all vehicles located at a facility. Solution process for (SPP-LRSP) which employs column generation may become more complicated with cutting planes with a certain structure. Since the structure of the cutting planes searched may effect the efficiency of the solution algorithm, the class of cutting planes considered may be more restrictive for (SPP-LRSP). However, one advantage of (SPP-LRSP) is that some cutting planes valid for (G-LRSP) are already implied by formulation (SPP-LRSP). This is evident from the Proposition 2.5 and the definition of the columns of (SPP-LRSP). This will further be investigated in Chapter 4.

2.5 Complexity

In terms of formal computational complexity, the LRSP is in the complexity class NP-hard. To show this, we consider an instance of the LRSP in which there is at most one vehicle at each facility, there are no facility and vehicle fixed costs, and the vehicle time limit is unrestricted. Such a special case is equivalent to an instance of the MDVRP, which was shown to be NP-hard by Lenstra and Kan (1981), and Bodin et al. (1983). The MDVRP aims to find a minimum-cost set of vehicle routes, each one starting from a depot and terminating at the same depot. Each customer must be served exactly once and there are specified lower and upper bounds for the

number of capacitated vehicles at each depot. In the created LRSP instance, location decision and facility costs are eliminated by setting the fixed cost of facilities to zero, and scheduling part is eliminated by setting the time limit to infinity. Solution of this instance results a set of routes with minimum total operating cost. This solution can easily be translated to a solution for the MDVRP. In the solution of the MDVRP, the allocation of the customers to the facilities, the vehicle routes and the total cost are the same as in the solution of the LRSP. Since we have shown that the MDVRP is a special case of the LRSP and the MDVRP is NP-hard, we can conclude that the LRSP is also NP-hard.

Chapter 3

Branch-and-Price Algorithm

In this chapter, we describe our solution methodology for the LRSP. Although it would be possible to develop a branch-and-cut algorithm for (G-LRSP), we suspect that such an approach would not yield positive results, given the inherent symmetry in the formulation, the weak LP relaxation bound and the known difficulty associated with solving the associated relaxations. Instead, we use the set partitioning-based formulation, which is similar to those that have been successful for other hard combinatorial problems. In our set partitioning-based formulation of the LRSP, each column corresponds to a feasible pairing. For medium- and large-scale instances, the number of feasible pairings can be extremely large, making it unlikely that we can efficiently solve instances that explicitly include all feasible pairings. We have therefore developed a branch-and-price algorithm in which we dynamically generate only a subset of the feasible pairings at each node of the tree, as is typical in such cases.

In the following sections, we first give an overview of the branch-and-price algorithm and describe some notation. We then describe the components of our branch-and-price algorithm. This involves describing the pricing problem and discussing solution approaches designed to solve the pricing problem, as well as describing the methods of branching used to partition the feasible region of the original problem. Finally, we present some implementation details including initialization, the column generation algorithm consisting of different pricing algorithms in combination,

upper bounding procedures and different versions of the branch-and-price algorithms providing exact and heuristic solutions for the LRSP.

3.1 Notation and Overview

Recall that branch and price is an approach that combines branch and bound with a column generation algorithm to solve the LP relaxations arising during the enumeration. We have outlined the main steps of a column generation algorithm (COLGEN) in Figure 1.4 and of a branch-and-price algorithm (BRANCH-PRICE) in Figure 1.5.

As noted before, we developed the branch-and-price algorithm for the set partitioning-based formulation of the LRSP. Because we have shown in Section 2.3 that adding both constraints (2.18) and (2.25) to (SPP-LRSP) strengthens the formulation, we use the following augmented (SPP-LRSP) formulation, denoted by (ASPP-LRSP) as the basis for our solution algorithm:

$$z^*_{\text{LRSP}} = \text{Min} \quad \sum_{j\in J} C^F_j t_j + \sum_{j\in J}\sum_{p\in P_j} C_p z_p \tag{3.1}$$

(ASPP-LRSP) s.t.

$$\sum_{j\in J}\sum_{p\in P_j} a_{ip} z_p = 1 \qquad \forall i \in I, \tag{3.2}$$

$$\sum_{p\in P_j}\sum_{i\in I} a_{ip} D_i z_p - L^F_j t_j \le 0 \qquad \forall j \in J,$$

$$\sum_{j\in J} t_j \ge R,$$

$$\sum_{p\in P_j} a_{ip} z_p - t_j \le 0 \qquad \forall j \in J, i \in I,$$

$$t_j \in \{0,1\} \quad \forall j \in J, \text{ and}$$

$$z_p \in \{0,1\} \quad \forall p \in P_j, j \in J.$$

In the remainder of this section, we provide an overview of the notation and the steps of the branch-and-price algorithm, BRANCH-PRICE (Figure 1.5). Recall that to initialize BRANCH-PRICE, we need to generate an initial set of columns to initialize the global column set denoted by

C_G. We describe the initialization approach to generating a subset of pairings in Section 3.4.1. The heuristic we apply for the initialization also provides an upper bound for z^*_{LRSP}. The formulation (ASPP-LRSP) is the original problem considered and corresponds to P^0 in the algorithm. The set of feasible solutions to (ASPP-LRSP) is denoted by S^0. Refer back to Chapter 1 for the notation used in this chapter.

The column generation algorithm for P^i begins by solving RMP_0^i and then, using the resulting dual solution information, formulates the pricing problem with the objective of either identifying new columns to add to RMP_0^i or proving optimality of the current solution. If the pricing algorithm identifies at least one column with negative reduced cost, then we add the identified columns to RMP_0^i and continue to iteratively re-optimize the current restricted LP relaxation and solve the pricing problem. If the pricing problem fails to find any column with negative reduced cost, the current solution is optimal for the LP relaxation of P^i which we refer to as z^i_{LP}. We describe the pricing problem and several solution approaches used in our column generation algorithm in Section 3.2. In Section 3.4, we present some implementation details of the branch-and-price algorithm for the LRSP. We present two variations of the branch-and-price algorithm that are used in our implementations.

Having briefly reviewed the branch-and-price algorithm for the LRSP, we discuss some challenges of the algorithm.

- The main challenge of applying column generation is solving the pricing problem, which is generally an integer programming model itself. One call to COLGEN (to obtain z^i_{LP} for some P^i) may require many calls to the oracle that solves the pricing problem. Therefore, an efficient solution method for the pricing problem is crucial for the efficiency of the whole algorithm. In order to improve the performance of COLGEN, one may opt to solve a more tractable relaxation of the pricing problem when possible. Replacing the original pricing problem with a relaxation yields a bound that is lower than it would otherwise. Alternatively, one can solve the pricing problem heuristically. Indeed, Step 3 of COLGEN does not require to find a column with the probably minimum cost in intermediate iterations. However, to

terminate the COLGEN in Step 3 (b) with a valid lower bound for the optimal LP relaxation value, one must find a column with the minimum cost, thus one must solve the pricing problem to optimality at least once in COLGEN.

- Another challenge related to implementing COLGEN is that solving RMP_k^i at any iteration k may become time consuming as $|C_G|$ gets larger. The column management may become a crucial part of the algorithm.

- In column generation algorithms, it is also very common to iterate between Steps 1 and 3 many times before termination. When there are alternative dual optimal solutions, the algorithm may jump many times from one primal LP solution to another without much improvement in the solution value.

- Besides the challenges inherent in the column generation algorithm, the main challenge in developing a branch-and-price algorithm is developing effective branching rules that do not destroy the structure of the pricing problem. At a non-root node of the tree, say node i, we have the subproblem P^i defined over the feasible set S^i which is obtained by applying terms of the branching disjunctions defining node i. The pricing problem at the root node is constructed based on the constraint set of P^0 in order to generate columns feasible for S^0. Therefore, the pricing problem constructed at the root node may need to be modified for S^i. Therefore, for an efficient algorithm, branching rules must not complicate the pricing problem and can easily be incorporated in the solution algorithm for the pricing problem.

In the following sections, we describe the components of the branch-and-price algorithm for the LRSP in more detail.

3.2 Pricing Problem

In this section, we describe the model-specific components of the column generation algorithm that is used in our branch-and-price algorithm. First, we describe the pricing problem in detail,

and then present various solution algorithms. Later in Section 3.4.2, we provide the algorithm-specific details of the overall column generation algorithm used in our branch-and-price algorithm.

3.2.1 Description

Formulation

The pricing problem aims to generate new feasible pairings, so the constraints are simply the linear inequalities defining the feasibility conditions for pairings presented in Section 2.1. As explained in Section 2.2.2, the set of pairings for each facility are independent, so we have a separate pricing problem for each facility. In iteration k in node i, the objective of the pricing problem is to obtain a pairing whose associated column has the smallest reduced cost with respect to the dual solution of RMP_k^i. To construct the objective function of the pricing problem for facility j, we consider the formulation P^i and obtain an expression for the reduced cost of a pairing. In this section, we describe the pricing problem and the solution algorithms used at the root node considering formulation (ASPP-LRSP). Later, when we describe branching disjunctions, we will discuss the necessary modifications to the pricing problem.

Let π_l be the dual variable associated with constraint (2.14) for customer l, μ_j be the dual variable associated with constraint (2.15) for facility j, and σ_{jl} be the dual variable for linking constraint (2.25) for facility j and customer l. For variable z_p for $p \in P_j$, the reduced cost $\hat{C}_p$ can be written as

$$\hat{C}_p = C_p - \sum_{l \in I} a_{lp}\pi_l + \sum_{l \in I} a_{lp} D_l \mu_j + \sum_{l \in I} a_{lp}\sigma_{jl}, \tag{3.3}$$

where C_p is the cost of pairing p. Thus, we can express the pricing problem for facility j as the problem of minimizing $\hat{C}_p$ subject to the previously detailed constraints (in Section 2.1) defining feasible pairings. A mathematical programming formulation of the problem for facility j is then

as follows:

$$\text{Min} \quad C^V + \sum_{i\in M}\sum_{k\in M} \hat{c}_{ik} x_{ik} \tag{3.4}$$

(SP$_p$) s.t.

$$\sum_{k\in M} x_{ik} \leq 1 \qquad \forall i \in I, \tag{3.5}$$

$$\sum_{k\in M} x_{ik} - \sum_{k\in M} x_{ki} = 0 \qquad \forall i \in M, \tag{3.6}$$

$$y_{ik} - L^V x_{ik} \leq 0 \qquad \forall i \in M, k \in M, \tag{3.7}$$

$$\sum_{k\in M} y_{ik} - \sum_{k\in M} y_{ki} + D_i \sum_{k\in M} x_{ik} = 0 \qquad \forall i \in I, \tag{3.8}$$

$$\sum_{i\in M}\sum_{k\in M} T_{ik} x_{ik} \leq L^T, \tag{3.9}$$

$$x_{ik} \in \{0,1\} \quad \forall i \in M, k \in M, \tag{3.10}$$

$$y_{ik} \geq 0 \qquad \forall i \in M, k \in M, \tag{3.11}$$

where $M = I \cup \{j\}$, $x_{ik} = 1$ if arc (i,k) is used in the solution and 0 otherwise; y_{ik} is equal to the amount of flow on arc (i,k); and $\hat{c}_{ik}$ is the *reduced cost* of arc (i,k) defined as follows.

$$\hat{c}_{ik} = \begin{cases} C^O T_{ik} - \pi_k + D_k \mu_j + \sigma_{jk} & \text{if } i \in M, \text{ and } k \in I, \\ C^O T_{ij} & \text{if } i \in I. \end{cases} \tag{3.12}$$

Recall the subproblem (SP$_j$) for facility j that is obtained when DWD is applied to (G-LRSP) (presented in Section 2.4). Notice that (SP$_p$) is equivalent to the problem that is obtained by decomposing (SP$_j$) by vehicle.

To get more insight into the objective function (3.4), note that in equation (3.3), the cost C_p is the sum of the fixed cost of a vehicle and the variable costs of the routes included in the pairing which are equal to the sum of the costs of the arcs in the route. We account for the additional terms $-\sum_{l\in I} a_{lp}\pi_l + \sum_{l\in I} a_{lp} D_l \mu_j + \sum_{l\in I} a_{lp}\sigma_{jl}$ from (3.3) in (3.4) by modifying the original arc costs of the arcs incoming to customer k by the values π_k, $D_k\mu_j$ and σ_{jk} associated with customer k. With this modification, the reduced cost of a pairing can be computed as the sum of

the reduced costs of the arcs in the pairing.

Elementary Shortest Path Problem With Resource Constraints (ESPPRC)

As we have just seen, the pricing problem (SP_p) can be formulated as a generic mixed integer programming model. By solving it this way, we do not take advantage of the special structure of the problem. Upon closer inspection, the pricing problem can be interpreted as an elementary shortest path problem with resource constraints (ESPPRC) on a modified network. In an ESPPRC, the objective is to find a shortest path between the source node and the sink node such that each other node is visited at most once and the resource constraints are not violated.

To properly represent our pricing problem as an ESPPRC, we need to construct a network, determine the resources, and determine the resource consumptions and costs of the arcs, so that a feasible path from the source node to the sink node corresponds to a feasible pairing that would be generated by solving (SP_p). To define the pricing problem for facility j, we construct a network with $|I|+2$ nodes: $|I|$ nodes for the customers, a node for facility j as the source node and a copy of facility j as the sink node. Let A^j be the arc set of the network for facility j. Then A^j consists of arcs from the source to customer nodes, arcs between customer nodes and the sink node, and arcs between pairs of customer nodes. Note that there are no arcs from customer nodes back to the source but there are arcs from the sink to customer nodes, this enables us to generate pairings including multiple routes when we generate paths from the source node to the sink node. In this network, in a path originating at the source node, an arc from the sink node to a customer node represents the beginning of a new route, whereas an arc from a customer node to the sink node represents the end of a route. Figure 3.1 shows an example network and a path on the network. In the example, the path corresponds to pairing (facility$_j$ - 3 - facility$_j$ - 1 - 2 - facility$_j$) which includes two routes.

We set the length and the cost of arc (i,k) to T_{ik} and $\hat{c}_{ik}$ (defined in (3.12)), respectively, for all $(i,k) \in A^j$. We let each customer node have a demand equal to its demand in the LRSP instance, the source have zero demand, and the sink have minus vehicle capacity demand. We use

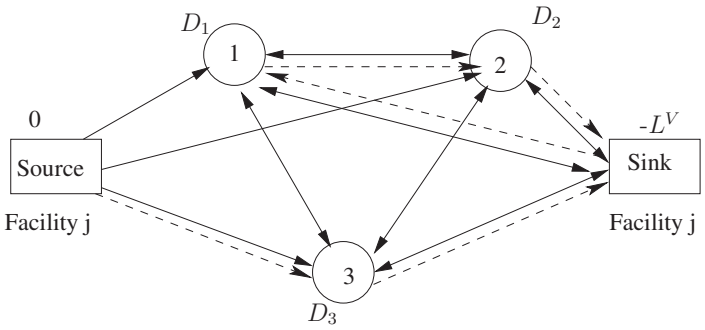


Figure 3.1: Example of a network and a path from source to sink (dashed line)

time and vehicle capacity as the two resources in the problem. As we construct a path, when we include an arc, the length of the arc and the demand of the destination node are used to update the elapsed time and the vehicle load (resources consumed), respectively. In addition, the total cost of the path is equal to the cost of the pairing associated with the path because we use the same cost parameter used in (SP_p). The process of checking the resource constraints and updating the resource consumption at each node will be discussed in detail in Section 3.2.2. In this network and with this design, an elementary resource constrained path from the source node to the sink node corresponds to a feasible pairing.

3.2.2 Exact Methods

Because the pricing problem can be formulated as an ESPPRC, we can draw on the extensive literature on this problem for potential solution algorithms. One particular special case occurs when the underlying network is acyclic. In this case the "elementary" property of the paths can be ignored, because in this case, any solution of the shortest path problem with resource constraints (SPPRC) is guaranteed to be feasible for the ESPPRC. Desrochers (1988) developed a label correcting algorithm for SPPRC based on dynamic programming that has a pseudo-polynomial complexity that depends on the size of the resource windows and the number of resources. The algorithm has been shown to be successful when there are tight resource constraints, but it becomes more time-consuming as the number of resources increases. As the algorithm builds partial

paths, it assigns a label to each, indicating the resource consumption. To eliminate the problem of an exponentially increasing number of labels, the algorithm applies dominance rules to delete dominated labels.

If the network has negative arc costs, which is generally the case in pricing problems, the elementary property of the paths must be considered for feasibility. Beasley and Christofides (1989) describe how to find elementary paths in a graph by adding an extra binary resource for each node showing whether the node is visited or not. However, this increases the number of resources and possible combinations exponentially. In fact, the ESPPRC was proved to be NP-Hard by Dror (1994). Feillet et al. (2004) adapt Desrochers's label correcting algorithm to find elementary paths and strengthen the algorithm by developing additional domination rules for the labels. We adapted Feillet et al. (2004)'s label correcting algorithm for our pricing problem.

We developed two solution approaches. One approach is to solve a single ESPPRC on the network described in Section 3.2.1. An alternative approach is to extend the first approach to a two-phase pricing problem in which each phase is also formulated and solved separately as an ESPPRC.

One-Phase Pricing Algorithm (1p-ESPPRC)

In this first approach, we use Feillet et al. (2004)'s algorithm to solve one ESPPRC on the constructed network. In the Feillet et al. (2004) algorithm, each path from the source to a node in the network is assigned a label, which is a vector of the reduced cost of that (partial) path, the resources consumed (vehicle load and elapsed time), the customer nodes included in the path, and the customer nodes that cannot be visited due to the resource constraints. Customers are considered *unreachable* if they have already been visited in the path or if they cannot be served by the vehicle without violating a resource constraint. The algorithm fans out from the source node, repeatedly examining the nodes of the graph. Each time it examines a node, the algorithm extends each of the paths to each possible successor node, continuing until no further extensions are feasible.

To explain our algorithm that is based on Feillet et al. (2004)'s algorithm more specifically, we let l_r^i be a label that corresponds to path r from source to node i. We use vector $l_r^i = (R_c^i, R_v^i, R_t^i, R_n^i, V_1^i, .., V_{|I|}^i)$ to store the information about path r where

$$
\begin{aligned}
R_c^i &= \text{reduced cost of the path (so far)},\\
R_v^i &= \text{total demand of the path (so far)},\\
R_t^i &= \text{time consumption of the path (so far)},\\
R_n^i &= \text{total number of unreachable customer nodes (so far)},\\
V_k^i &= \left\{ \begin{array}{ll}
-2 & \text{if customer } k \text{ is temporarily unreachable for the path}\\
-1 & \text{if customer } k \text{ is unreachable for the path because of the resource limits,}\\
0 & \text{if customer } k \text{ is not in the path and it is reachable for the path with the resources,}\\
s > 0 & \text{if customer } k \text{ is the } s^{th} \text{ node in the path,}
\end{array} \right\} \forall k \in I.
\end{aligned}
$$

Note that the dimension of the vector V is $|I|$, so that there is one member for each customer node in the network. In label l_r^i, if customer k is visited in the path, V_k^i keeps the place of the customer in the path and remarks that customer k is unreachable for this label because of the elementary property. In Feillet et al.'s algorithm, if customer k is not visited in the path r, V_k^i is set to either 0 or -1. Even though we have two resource constraints (vehicle capacity and time limit) in the problem, we only check the time limit constraint to mark a customer as "unreachable", i.e., to set $V_k^i = -1$, because if vehicle load exceeds vehicle capacity by adding customer k to the path, that does not mean that we cannot add customer k to this path in the future. It may be possible that a path extended from path r stops at the sink node and would have enough capacity to visit customer k in further extensions. We explain the condition when V_k^i is set to -2 (which is used to define a new domination rule) later in this section.

We do not need an element in V for the source, because every path starts from the source and can never return to it. Similarly, we do not need an element for the sink, because the sink must always be reachable for every path. Extending a path to the sink is always resource feasible because (1) demand of the sink is the negation of vehicle capacity (and hence total demand would be less than the vehicle capacity), and (2) while constructing label r associated with a non sink node i ($l_r^i = (R_c^i, R_v^i, R_t^i, R_n^i, V)$), the time limit constraint is checked to ensure that the path is further extendable to the sink, i.e., $R_t^i + T_{i,sink} \leq L^T$.

Our algorithm based on the Feillet et al.'s algorithm for solving the ESPPRC is referred to as 1p-ESPPRC and includes three main parts:

1p-ESPPRC:

1. **Initialization step**

   (a) To begin the algorithm, a label list is created for each node to keep the generated labels (partial paths to the node).

   (b) All label lists except the list for the source node are initially empty. The source has one label, which is a zero vector, i.e., resource consumptions and the reduced cost are zero, and all customers are reachable.

2. **Extension step**

   (a) For each node, the label list is searched and all unprocessed labels are extended to all possible successors of the node. Unprocessed means that the label was not extended in previous iterations. The label l_r^i for node i can be extended to node k if

      i. node k is reachable: $k = sink$ or $V_k^i = 0$, and

      ii. the path is resource feasible: $R_v^i + D_k \leq L^V$ and $R_t^i + T_{i,k} + T_{k,sink} \leq L^T$.

   (b) If it is feasible to extend l_r^i to node k, a new label is created for node k by updating the

resource consumptions. Let l_p^k be the label extended from l_r^i. Then

$$
\begin{aligned}
R_c^k &= R_c^i + \hat{c}_{ik} \\
R_v^k &= \max\{R_v^i + D_k, 0\}, \\
R_t^k &= R_t^i + T_{i,k}, \\
V_g^k &= \left\{ \begin{array}{ll} N & \text{if } g = k \text{ and } N = \#\text{ of nodes in the path,} \\ -1 & \text{if } V_g^i = 0 \text{ and } R_t^k + T_{k,g} + T_{g,sink} > L^T, \\ V_g^i & \text{if } V_g^i \neq 0, \\ 0 & \text{otherwise,} \end{array} \right\} \forall g \in I, \\
R_n^k &= \sum_{g \in I : V_g^i \neq 0} 1.
\end{aligned}
$$

(c) Mark the labels that were extended as processed.

3. **Elimination step**

(a) After extending unprocessed labels of a node to its successors, and constructing new labels for the successors, these new labels are combined with their previously created label lists. Let node i be a node with new labels. To combine the label list of i with the list of new labels created for i, each label is compared with the others in the list and only the *non-dominated labels* are retained. Let l_1^i and l_2^i be two labels for node i. Label l_1^i dominates label l_2^i if all of the following conditions are satisfied:

i. the current travel time and the vehicle load of l_1^i are less than or equal to those of l_2^i,

ii. the reduced cost of l_1^i is less than or equal to the reduced cost of l_2^i, and

iii. the number of total unreachable customer nodes of l_1^i are less than or equal to those of l_2^i, and for all nodes $g \in I$, g is reachable only in label l_1^i or g is unreachable in both labels l_1^i and l_2^i.

All dominated labels are deleted from the label lists of the nodes.

(b) If new labels were added to the list of any node after the domination algorithm was applied, then the algorithm returns to the extension step. When all the labels are extended and no new labels are added to the lists of nodes, the algorithm terminates.

Additional Domination Rule. We extended Feillet et al.'s algorithm by adding a new domination rule. Executing the described 1p-ESPPRC on the constructed network leads to paths from the source to the sink that correspond to pairings including sets of *ordered* vehicle routes. Because the ordering of the routes within a pairing is arbitrary, we would like to avoid generating multiple paths that are different permutations of a set of routes and correspond to the same pairing. For example, a call to 1p-ESPPRC may generate some or all of the following paths:

- (source - 1 - 2 - sink - 3 - 4 - sink - 5 - 6 - sink),
- (source - 1 - 2 - sink - 5 - 6 - sink - 3 - 4 - sink),
- (source - 3 - 4 - sink - 1 - 2 - sink - 5 - 6 - sink),
- (source - 3 - 4 - sink - 5 - 6 - sink - 1 - 2 - sink),
- (source - 5 - 6 - sink - 1 - 2 - sink - 3 - 4 - sink), and
- (source - 5 - 6 - sink - 3 - 4 - sink - 1 - 4 - sink).

However, those six paths are equivalent with respect to total cost and time, and they correspond to the same pairing composed of the following three routes: (facility - 1 - 2 - facility), (facility - 3 - 4 - facility) and (facility - 5 - 7 - facility). To eliminate generation of all possible ordered sets, we introduce a new domination rule in which we force a specific order for routes in the pairing, which decreases the state space of the algorithm significantly. We add the following condition to the algorithm.

Rule The index of the first customer node of each route should be less than the index of the first customer node in the following route.

The modified algorithm only generates paths from source to sink in which the first node of each route is less than the first node of the following route. Thus, the algorithm with this rule generates only path (source - 1 - 2 - sink - 3 - 4 - sink - 5 - 6 - sink).

To apply this rule, we add variable R_f^i to keep the index of the first customer node in the current route in the path to the definition of label l_r^i, where $l_r^i = (R_c^i, R_v^i, R_t^i, R_n^i, R_f^i, V_1^i, .., V_{|I|}^i)$. We make the following changes in 1p-ESPPRC:

- *In the extension step of the algorithm*, in part (b), while constructing label l_p^k from l_r^i, we set $R_f^k = k$ if i is the source or the sink (k will be the first node in the new route), and $R_f^k = R_f^i$ (still in the same route, thus the first node is the same) otherwise. We also need to modify the procedure for updating vector V^k.

$$V_g^k = \left\{ \begin{array}{ll} N & \text{if } g = k \text{ and } N = \#\text{ of nodes in the path,} \\ V_g^i & \text{if } V_g^i \neq 0 \text{ and } V_g^i \neq -2, \\ -1 & \text{if } V_g^i = 0 \text{ or } V_g^i = -2, \text{ and } R_t^k + T_{k,g} + T_{g,sink} > L^T, \\ -2 & \text{if } k \text{ is sink, } V_g^i = 0 \text{ and } g < R_f^k, \\ 0 & \text{otherwise,} \end{array} \right\} \forall g \in I,$$

$$R_n^k = \sum_{g \in I: V_g^i \neq 0 \text{ or } V_g^i \neq -2} 1.$$

  Here, $V_g^k = -2$ means that node g is *temporarily unreachable* because, based on the new rule, it cannot be the first node in the new route. However, it may become part of the path during further extensions. Therefore, when we calculate number of unreachable nodes R_n^k, we do not count temporarily unreachable nodes. We treat them as reachable nodes. In addition, while extending a path to other nodes after the first node in the route, we change the condition temporarily unreachable to reachable by setting V_g^k to 0.

- *In the elimination step of the algorithm*, the steps and the conditions that should be satisfied to decide dominated labels are the same. While executing condition (iii) in part (a), if there

are temporarily unreachable nodes, we count them as reachable. The only modification to this step occurs when each label dominates the other. In the previous version, we were allowed to pick either one of l_1^i or l_2^i arbitrarily and delete the other one. However, in this version, if l_1^i dominates l_2^i and l_2^i dominates l_1^i based on conditions (i), (ii) and (iii), we keep the one with smallest R_f^i (first node in the current route).

At the end of a pricing algorithm, each label at the sink node corresponds to a feasible pairing with an associated reduced cost. The costs of paths at the sink node are updated with the vehicle fixed cost that must be included in the reduced cost of a variable, and any path with negative reduced cost is a candidate pairing to add to RMP$_k^0$ at iteration k. If the algorithm concludes with no negative reduced cost labels on the sink node, then we can conclude that the RMP$_k^0$ is optimal and the solution value is equal to the z_{LP}^0.

Two-Phase Pricing Algorithm (2p-ESPPR)

An alternative approach to solving the exact pricing problem is to extend the first approach (1p-ESPPRC) to a two-phase pricing problem: first, generate a set of feasible vehicle routes and then combine them to construct a feasible pairing of minimum reduced cost. Each phase is also formulated as a network problem and solved as an ESPPRC. Since a pairing is a set of routes, we can express the variable part of the reduced cost of a pairing as the sum of the reduced costs of the routes forming the pairing. For there to exist a negative reduced cost composed of more than one route, there must exist routes with negative reduced cost. Otherwise, the pairing including only the single route with the lowest reduced cost would be the lowest cost pairing overall.

Phase One: Generating Routes. To generate only feasible routes instead of pairings, we solve a regular ESPPRC problem on the constructed network with $|I| + 2$ nodes. We delete the arcs between the sink and the customers and terminate each path at the sink and we use algorithm 1p-ESPPRC without the additional domination rule. At the end of the algorithm, each label at the sink node corresponds to a feasible vehicle route with an associated reduced cost.

Let L_S denote the set of feasible routes with negative reduced cost at the sink. If L_S is non-empty, we continue with phase two. Otherwise, we add the fixed cost from expression (3.3) to the cost of the smallest reduced cost route to obtain the reduced cost of that single-route pairing. If this reduced cost is negative, then we have a pairing that we can add to the restricted model. Note the implication that, even in the case where there are no *routes* with negative reduced cost, we may still get a negative reduced cost *pairing* (consisting of a single route) when the fixed cost associated with the vehicle is negative. This can occur after applying certain branching rules (see Section 3.3.1).

Phase Two: Generating Pairings. The objective of phase two is to generate feasible pairings with negative reduced cost by combining the routes in set L_S. Two (or more) vehicle routes can be combined to form a feasible pairing if they do not have customer nodes in common and they can both (all) be completed within the vehicle's time limit. Finding a feasible pairing with the minimum reduced cost again corresponds to solving an ESPPRC with a single resource constraint (time).

We construct a network with $2+|L_S|$ nodes: a source node, a sink node and one "intermediate" node to represent each of the vehicle routes in set L_S. The network includes an arc from the source node to each intermediate node, arcs between pairs of intermediate nodes, and an arc from each intermediate node to the sink node. Note that, because the sequence of routes in a pairing is arbitrary, we only include an arc from an intermediate node to another intermediate node with higher index. Associated with each arc inbound to an intermediate node are two values: the time required to complete the corresponding route and the reduced cost of the route. The time and cost for arcs inbound to the sink are zero. To find a feasible pairing with the minimum reduced cost, we apply 1p-ESPPRC. The costs of pairings at the sink node are updated with the fixed cost that must be included in the reduced cost of a variable, and any pairing with negative reduced cost is a candidate pairing to add to the restricted model in the column generation.

Since we solve the exact ESPPRC algorithm using 1p-ESPPRC in each phase of the algorithm,

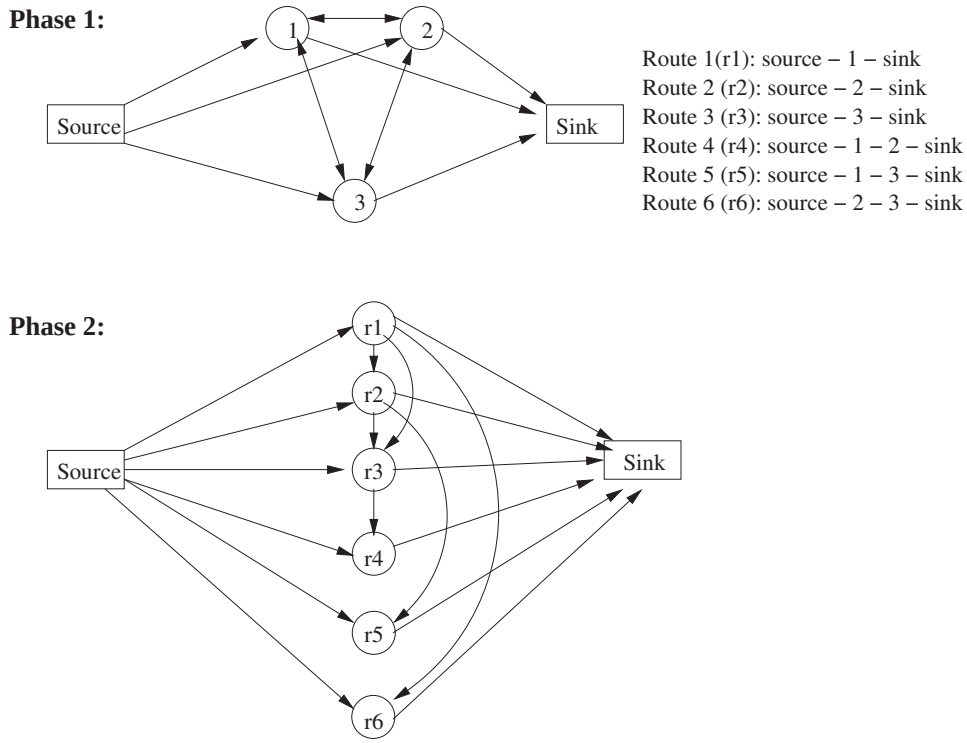


Figure 3.2: Demonstration of Phase 1 and Phase 2 Networks on a Small Example

the overall pricing algorithm is exact and results in a column with the most negative reduced cost, if any. We refer to this exact pricing algorithm as 2p-ESPPRC. To clarify the algorithm, we demonstrate how we construct the networks for each phase in Figure 3.2.

Generation of Multiple Pairings

In general, at the completion of 1p-ESPPRC or the second phase of 2p-ESPPRC, there is more than one label at the sink. We may have a set of pairings with negative reduced cost that includes a pairing with the most negative cost. Therefore, we may add multiple columns to the RMP$_k^i$ at iteration k of the column generation algorithm for node i. If we want to add at most n columns in any iteration, we can sort the labels on the sink node in ascending order based on the reduced cost and choose the first n or less number of labels with negative reduced cost in order to add to RMP$_k^i$.

In Section 5.4.1, we present computational results comparing the performance of 1p-ESPPRC, 1p-ESPPRC with the additional domination rule and 2p-ESPPRC, and computational results evaluating the effect of adding multiple columns instead of a single column to the performance of the column generation algorithm.

3.2.3 Heuristic Methods

An exact algorithm is guaranteed to generate a column with the minimum reduced cost. The downside of the approach is that it may not be very efficient for large problems. As the number of customers, the vehicle capacity and the time limit increase, the state space of the algorithm, i.e., the number of feasible partial paths, also increases, so the number of labels in the label list of each node in the algorithm increases. Thus, processing all labels and terminating the algorithm takes substantially more time.

A common approach in the literature to overcoming this problem is to use heuristics to supplement the exact procedure for the pricing problem to improve the efficiency of a branch-and-price algorithm. To find feasible pairings more quickly, here we define two heuristic versions of a labeling algorithm that we use in our solution algorithm. We can use these heuristic modifications for both 1p-ESPPRC and 2p-ESPPRC.

ESPPRC with Label Limit (ESPPRC-LL)

This heuristic version was first proposed by Dumitrescu (2002) and restricts the state space of the exact algorithm by directly restricting the number of labels. In particular, during 1p-ESPPRC, or either phase of 2p-ESPPRC, we limit the maximum number of non-processed labels allowed at each node. During the algorithm, as we update the list of labels in the elimination step, at the end of the domination algorithm, we order the non-dominated labels in *increasing order of reduced cost* and we keep at most the first *label limit* number of non-processed labels. We refer to these heuristic versions of pricing algorithms as 1p-ESPPRC-LL(l) and 2p-ESPPRC-LL(l), where l is the value of the label limit. For small label limits, the algorithm terminates very quickly. And as l

gets larger, the heuristic version approaches to the exact algorithm.

ESPPRC for a Subset of Customers (ESPPRC-CS)

This second heuristic restricts the state space by considering only a subset of the customers. In this way, the total number of labels to be generated is restricted in advance. In each pricing iteration for a facility, we determine the *two customers* that have the *lowest reduced costs from the facility*. Let $j \in J$ be the facility index, we choose customers i_1 and i_2 such that $\hat{c}_{ji_1} \leq \hat{c}_{ji_2} \leq \hat{c}_{ji}$ for all $i \in I \backslash \{i_1, i_2\}$. Let $I_u = \{i_1, i_2\}$ be the subset. We initialize the subset with these two customers, since we expect to have at least two routes in a pairing. Then, by exploring the arcs from the subset of customers to the remaining customers, we repeatedly add one customer that is connected by a cheapest arc to the subset until the size of the subset is equal to the determined *subset size*. Particularly, we let $I_u = I_u \cup \{k\}$ where $k = \text{argmin}_{k \in I \backslash I_u} \{\text{min}_{i \in I_u} \{\hat{c}_{ik}\}\}$ We stop when $|I_u| = u$ where u is the determined subset size.

After we determine I_u, we run the exact pricing algorithm with this smaller customer set. We refer to these heuristic versions of pricing algorithms as 1p-ESPPRC-CS(u) and 2p-ESPPRC-CS(u), where u is the subset size. In 2p-ESPPRC-CS(u), we use the selected customers in Phase 1, while the state space of Phase 2 reduced as a result of the reduction in the state space of Phase 1.

If a heuristic version of the pricing algorithm identifies columns with negative reduced cost, they are feasible and can be added to RMP_k^0. However, these columns are not necessarily the columns with most negative reduced cost for a given dual vector. Therefore, if a heuristic version of the pricing algorithm does not identify any columns with negative reduced cost, the exact pricing algorithm is applied at least once in order to conclude whether or not there exist columns with negative reduced cost or to evaluate a lower bound for z_{LP}^0. In Section 5.4.2, we present some computational results evaluating the effect of applying heuristic versions of the pricing algorithm in addition to the exact pricing algorithm and comparing the performance of both heuristic approaches.

3.3 Branching

An important step in developing a branch-and-price algorithm is devising good branching rules for partitioning the original feasible solution set. In this section, we first list the disjunctions used to partition S^0 in our branch-and-price algorithm. We describe how these disjunctions are imposed to produce new subproblems and how the pricing problem is affected by the imposition of these disjunctions. Because the disjunctions applied may affect the structure of the pricing problem and may make the pricing algorithm more difficult, our main objective is to find disjunctions that eliminate as much of the integer infeasible region as possible but that minimize the increase in difficulty of solving the pricing problem. We refer to the disjunctions used as branching rules or branching decisions. Second, we discuss the completeness of the branching rules in order to provide z^*_{LRSP} at the termination of the branch-and-price algorithm.

3.3.1 Methods

We apply four branching rules in our branch-and-price algorithm:

1. **Branching on the facility location variables t:** At any node of the branch-and-price tree, if the corresponding LP solution has at least one fractional facility location variable, then we apply standard variable dichotomy branching for the facility location variables. We select one of the fractional facility location variables, say t_j, as our branching variable.

    - In the left node, we set $t_j = 1$, which forces facility location j to be open. The pricing problem for the corresponding node does not change, since variable t is not present in the constraints of the pricing problem (SP$_p$), and the pricing problem for each facility is independent from the others.
    - In the right node, we set $t_j = 0$, which forces facility j to be closed. Since the facility is closed, no pairings associated with the facility are allowed in the solution, so there is no need to solve the pricing problem. In addition, we set $z_p = 0 \; \forall p \in P_j$.

2. **Branching on the total number of vehicles at each facility:** By definition, a pairing is associated with a single facility and corresponds to one vehicle. Thus, in any integer solution, the number of vehicles (pairings) used at any facility must be integer. Let w_j be the number of vehicles used at facility j. Then,

$$w_j = \sum_{p \in P_j} z_p \qquad \forall j \in J. \tag{3.13}$$

If we add the equalities (3.13) to (ASPP-LRSP), we can branch explicitly on the w_j variables. Assume that in a fractional solution $w_j = b$, where b is not integer for some $j \in J$. We create two new nodes:

$$\begin{aligned} w_j &\leq \lfloor b \rfloor \qquad \text{(the left node)}, \\ w_j &\geq \lceil b \rceil \qquad \text{(the right node)}. \end{aligned}$$

Branching on w_j instead of implicitly branching on $\sum_{p \in P_j} z_p$ is an example of explicit constraint branching, an idea developed by Appleget and Wood (2000) to improve the performance of branch-and-bound algorithms for mixed integer linear problems. These new variables make it easier to identify the necessary updates to the pricing problem. Let β_j be the dual variable associated with constraint (3.13) for facility j. Then, the reduced cost of the variable z_p changes to the following:

$$\hat{C}_p = C_p - \sum_{i \in I} a_{ip}\pi_i + \sum_{i \in I} a_{ip} D_i \mu_j + \sum_{i \in I} a_{ip}\sigma_{ji} - \beta_j. \tag{3.14}$$

In equation (3.14), we can interpret β_j as a fixed cost for facility j that is independent of pairing p and the included customers. Therefore, we do not need to change our network structure or the pricing problem structure to incorporate β_j. Instead, we can calculate the cost of a path in the modified network in the same way as before and then add the fixed cost

$-\beta_j$ to the cost of the solution to calculate the correct reduced cost of the column.

3. **Branching on assignment of a customer to a specific facility:** At any node of the branch-and-price tree, if the solution is fractional and Rules 1 and 2 are not applicable, then we consider the assignment of a customer to a specific facility. In a fractional solution, a customer may be partially served by two or more pairings assigned to different facilities. In this case, we can create a branching rule to force a customer to be assigned to a specific facility. To do so, for a customer k served by more than one facility and a facility j such that $\sum_{p \in P_j} a_{kp} z_p$ is fractional, we create two nodes.

   - In the right node (say node i), we force customer node k to be served by facility j by adding the inequality

   $$\sum_{p \in P_j} a_{kp} z_p \geq 1 \tag{3.15}$$

   to P^i. Let γ_{jk} be the dual variable associated with inequality (3.15) in RMP_0^i. To incorporate this into the pricing problem for facility j, we modify the reduced cost of any arc entering node k with $-\gamma_{jk}$ (i.e., $\hat{c}_{lk} = \hat{c}_{lk} - \gamma_{jk}\ \forall l \in M$) and solve the pricing problem as before. In addition, we can remove customer k from the network for the pricing problems of other facilities.

   - In the left node (say node $i+1$), we forbid the assignment of customer node k to facility j by adding to P^{i+1} the constraint

   $$\sum_{p \in P_j} a_{kp} z_p \leq 0.$$

   To incorporate this restriction into the pricing problem for facility j, we remove node k from the network of facility j. The pricing problems for the other facilities do not change.

4. **Branching on flow between pairs of customer nodes:** At any node of the branch-and-price tree, if the solution is fractional and Rules 1, 2 and 3 are not applicable, then we may encounter the case where a customer is covered by two or more (fractional) pairings associated with the same facility. Then, we branch on flows on single arcs between pairs of customer nodes using a modified version of the Ryan and Foster (1981) branching rule first suggested by Desrochers and Soumis (1989). In the original Ryan and Foster (1981) branching rule, a pair of set partitioning constraints is selected. In one branch, the rule is to branch on the sum of the variables that satisfy both constraints and, in the other branch, the rule is to branch on the sum of the variables that do not satisfy both constraints simultaneously. In the VRP context, set partitioning constraints are associated with customers, and Desrochers and Soumis (1989) modify the Ryan and Foster branching rule by looking only at the variables in which the selected pair of customers are in consecutive order or not. This modification makes it possible to incorporate this branching rule to the pricing problem without changing the structure or the solution method.

   To apply the modified Ryan and Foster branching rule, we select a pair of customer nodes (n_1, n_2) such that

$$0 < \sum_{j \in J} \sum_{\{p \in P_j | (n_1 \rightarrow n_2) \in p\}} z_p < 1, \tag{3.16}$$

   where $\{p \in P_j | (n_1 \rightarrow n_2) \in p\}$ is the set of pairings (of facility j) in which n_1 is followed directly by n_2 for some $n_1, n_2 \in I$.

   In one branch, we require that the flow on the directed arc between the selected pair of customers is one in all possible solutions. In the other branch, we forbid the flow on the

directed arc between the selected pair of customers. We then create two new nodes:

$$\sum_{j \in J} \sum_{\{p \in P_j | (n_1 \rightarrow n_2) \in p\}} z_p \geq 1 \quad \text{(the left node)},$$
$$\sum_{j \in J} \sum_{\{p \in P_j | (n_1 \rightarrow n_2) \in p\}} z_p \leq 0 \quad \text{(the right node)}.$$

- In the left node (say node i), we are specifying that either node n_1 is followed by node n_2 in a pairing or that neither node appears. To enforce this rule for the pairings to be generated, we update the arcs in the network for the pricing problem. In the pricing problem, all arcs leaving node n_1 and all arcs entering node n_2 are deleted except the arc (n_1, n_2). Therefore, if a pairing includes one of the nodes, then it includes the arc (n_1, n_2). To delete an arc in the network, we set the cost of the arc to infinity. In the RMP$_0^i$, to satisfy the branching rule for the previously created pairings, we set to zero the variables corresponding to pairings that include only one of the nodes, n_1 or n_2, or that include both nodes but in which n_2 does not immediately follow n_1.
- In the right node (say node $i+1$), we are specifying that n_2 cannot follow n_1 in a feasible pairing. The feasible pairings may include n_1 or n_2 only or may include both of them as long as n_2 does not follow n_1 directly. To apply this rule for the pairings to be generated, we simply delete the arc (n_1, n_2) from the network in the pricing problem. In the RMP$_0^{i+1}$, we set to zero the previously created pairings in which n_1 is followed by n_2.

These four branching rules guarantee that the branch-and-price algorithm terminates with z^*_{LRSP}. We further investigate this statement in the next section.

3.3.2 Completeness

In a standard branch-and-price algorithm, nodes satisfying some conditions can be pruned. For each node i in the tree, Step 3 of the BRANCH-PRICE (Figure 1.5) checks these conditions which

are:

1. Is $z^i_{LP} \geq z_{UB}$?

2. Is the LP relaxation solution associated with z^i_{LP} (denoted by $(\bar{t}^i, \bar{w}^i, \bar{z}^i)$) feasible for P^0?

If node i satisfies one of these conditions, it can be pruned. In our branch-and-price algorithm, there may exist some nodes that satisfy none of the conditions listed above, but to which none of the four branching rules described in Section 3.3.1 is applicable. This issue brings to question whether the branching rules listed in the previous section are complete. Does our branch-and-price algorithm for the LRSP always guarantee an optimal solution?

In our branch-and-price algorithm, if node i cannot be pruned based on the conditions (1) and (2) and none of the branching rules is applicable, then node i satisfies the following:

i. $(\bar{t}^i, \bar{w}^i, \bar{z}^i)$ is feasible but not integer, and $z^i_{LP} < z_{UB}$,

ii. $\bar{t}^i \in \mathbb{Z}^{|J|}$,

iii. $\bar{w}^i \in \mathbb{Z}^{|J|}$,

iv. $\sum_{p \in \bar{P}_j} a_{kp} \bar{z}^i_p \in \mathbb{Z}\ \forall j \in J, k \in I$, where $\bar{P}_j$ is the set of pairings for $j \in J$ in C_G, and

v. $\sum_{j \in J} \sum_{\{p \in \bar{P}_j | (n_1 \rightarrow n_2) \in p\}} \bar{z}^i_p \in \mathbb{Z}\ \forall n_1, n_2 \in I$.

We will show that the nodes satisfying these conditions can still be fathomed based on fathoming condition (2). We can construct an integer solution from $(\bar{t}^i, \bar{w}^i, \bar{z}^i)$ with the same objective function value and may update z_{UB}. Therefore, we will conclude that our branching rules are complete.

In the LP solution vector satisfying conditions (ii) to (v), each open facility is fully utilized and the set of customers is divided into disjoint non-empty subsets, each of which is assigned to a single open facility. Therefore, $(\bar{t}^i, \bar{w}^i, \bar{z}^i)$ can be decomposed into smaller solution vectors, one for each facility. Let $(\bar{t}^i_j, \bar{w}^i_j, \bar{z}^i_j)$ be the vector associated with facility $j \in J$. Each of these vectors corresponds to a solution to a single facility LRSP instance and thus can be investigated separately.

Based on this observation, we investigate the solution vector associated with a single facility. The statements are true for a solution vector corresponding to any facility. In a fractional $(\bar{t}_j^i, \bar{w}_j^i, \bar{z}_j^i)$, because $\bar{t}_j^i$ and $\bar{w}_j^i$ are integer, at least two pairing variables must be fractional. By modifying the solution values based on the structure of the pairings associated with fractional variables, we can change this solution to an integer solution associated with the same facility. To explain how, we will show that the pairings associated with the fractional variables satisfying conditions (ii) - (v) have a certain structure.

Let r_1 and r_2 be two routes that can be included in a feasible pairing, and let I_1 and I_2 be the set of customers visited by routes r_1 and r_2, respectively.

Definition 3.1 Node disjoint: *Routes r_1 and r_2 are node disjoint if $I_1 \cap I_2 = \emptyset$.*

Definition 3.2 Identical (Same): *Routes r_1 and r_2 are identical if they both include exactly the same set of customers as well as the same set of arcs between these customers.*

We can represent a pairing by specifying the combination of routes included in the pairing. The routes included in a pairing are node disjoint by construction because in a pairing, a customer may be visited at most once. However, in a non-integer solution vector for a facility, at least two variables associated with pairings are fractional and thus one or more customer nodes may be included in more than one pairing. And, in fact, at least two pairings corresponding to fractional variables must include either the same route(s), different but non-node disjoint routes, or both. We illustrate this result using an example. Solution 1 is an example of a fractional solution (for a single facility) that includes pairings that share only the identical routes and all other routes are node disjoint. Solution 2 is an example of a fractional solution that includes pairings that have non-node disjoint routes.

Example Solution 1: Same or node disjoint routes:

$$z_1 = 0.5 \quad : \quad \text{facility}_1 - 1 - 2 - \text{facility}_1$$

$$z_2 = 0.5 \quad : \quad \text{facility}_1 - 1 - 2 - \text{facility}_1 - 3 - 4 - 5 - \text{facility}_1$$

$$z_3 = 0.5 \quad : \quad \text{facility}_1 - 6 - 7 - \text{facility}_1 - 3 - 4 - 5 - \text{facility}_1$$

$$z_4 = 0.5 \quad : \quad \text{facility}_1 - 6 - 7 - \text{facility}_1$$

Solution 2: Different and non-disjoint paths:

$$z_1 = 0.5 \quad : \quad \text{facility}_1 - 1 - 2 - \text{facility}_1 - 3 - 4 - 5 - \text{facility}_1$$

$$z_2 = 0.5 \quad : \quad \text{facility}_1 - 4 - 6 - 7 - \text{facility}_1$$

$$z_3 = 0.5 \quad : \quad \text{facility}_1 - 5 - \text{facility}_1 - 7 - 1 - 2 - 3 - \text{facility}_1$$

Both solution 1 and 2 are fractional since at least two variables associated with pairings including the same customers are selected. However, in solution 1, selected pairings not only include the same customers but also include the same routes such as route facility$_1$ - 1 - 2 - facility$_1$. On the other hand, in solution 2, pairings do not include the same routes but there are at least two pairings (such as pairings 1 and 2) that include non-node disjoint routes.

A fractional solution for a facility may be similar to either solution 1 or solution 2, or it can be a combination of these two. However, for a solution vector satisfying conditions (i) - (v) we can state the following result.

Lemma 3.1 *In an LP relaxation solution vector associated with node i (say $(\bar{t}^i, \bar{w}^i, \bar{z}^i)$) in the branch-and-price algorithm and satisfying conditions (i) - (v), there exists at least one facility with an associated fractional solution vector. Furthermore, a fractional solution vector for such facility corresponds to pairings including identical or node disjoint routes (as in solution 1). At least two pairings have at least one identical route. Finally, all non identical routes in the pairings*

are node disjoint.

Proof We have noted before that $(\bar{t}^i, \bar{w}^i, \bar{z}^i)$ can be decomposed by facility because of conditions (ii) - (iv). There must exist at least one facility (say j') with an associated fractional solution vector because of condition (i). Let $(\bar{t}^i_{j'}, \bar{w}^i_{j'}, \bar{z}^i_{j'})$ be the fraction solution vector corresponding to facility j'. Because of conditions (ii) and (iii), there must be at least two fractional pairing variables (say $\bar{z}^i_{j'1}$ and $\bar{z}^i_{j'2}$), and thus there must exist at least one customer (say k) visited by both pairings.

Let r_1 and r_2 be the routes visiting customer k in pairings represented by $\bar{z}^i_{j'1}$ and $\bar{z}^i_{j'2}$, respectively. Because r_1 and r_2 are both visiting customer k, they must be either the same route or they are not node disjoint. If they are not identical and not node disjoint, then there exists at least one pair of customers n_1 and n_2 such that arc (n_1, n_2) is either in route r_1 or in r_2, but not in both. However, this contradicts with the satisfaction of condition (v). Therefore, r_1 and r_2 must include the same set of arcs and thus be identical. Therefore, if two fractional pairings visit the same customers, they have to include exactly the same route(s) covering the same customers. This concludes that a fractional set of pairings must have at least one route included in at least two pairings, and non identical routes must be node disjoint. □

Lemma 3.2 *If $(\bar{t}^i, \bar{w}^i, \bar{z}^i)$ satisfies conditions (i) - (v), then there is an integer solution of the same value.*

Proof Let $(\bar{t}^i_j, \bar{w}^i_j, \bar{z}^i_j)$ be the vector obtained from $(\bar{t}^i, \bar{w}^i, \bar{z}^i)$ and associated with facility $j \in J$. Based on Lemma 3.1, the pairings corresponding to the fractional pairing variables in $(\bar{t}^i_j, \bar{w}^i_j, \bar{z}^i_j)$ include only identical routes and node disjoint routes $\forall j \in J$. By eliminating identical routes from all but one of the pairings, we can obtain a set of pairings that includes each route exactly once. This set of pairings corresponds to an integer solution. The objective function value of the new solution is the same with $(\bar{t}^i, \bar{w}^i, \bar{z}^i)$ because both solutions have exactly the same set of node disjoint routes and the flow on each node is 1. □

Therefore, based on Lemma 3.2, by setting the LP solution value equal to the value of an incumbent solution, we can fathom a node satisfying conditions (i) - (v) based on condition (2).

This proves the following statement.

Proposition 3.1 *The branching rules listed in Section 3.3.1 are complete and guarantee that the branch-and-price algorithm provides an optimal solution.*

We conclude this section with the following example of an integer solution that can be obtained from solution 1 which satisfies conditions (i) - (v).

Example Solution 3: Integer solution from solution 1 based on Lemma 3.2:

$$
\begin{aligned}
z_1 = 0 \quad &: \quad \text{facility}_1 - 1 - 2 - \text{facility}_1 \\
z_2 = 1 \quad &: \quad \text{facility}_1 - 1 - 2 - \text{facility}_1 - 3 - 4 - 5 - \text{facility}_1 \\
z_3 = 0 \quad &: \quad \text{facility}_1 - 6 - 7 - \text{facility}_1 - 3 - 4 - 5 - \text{facility}_1 \\
z_4 = 1 \quad &: \quad \text{facility}_1 - 6 - 7 - \text{facility}_1
\end{aligned}
$$

3.4 Implementation Details

In this section, we discuss some implementation details for the algorithm BRANCH-PRICE designed to solve the LRSP.

3.4.1 Initialization

To initialize BRANCH-PRICE, we need to initialize the column set C_G including location variables and an initial set of pairing variables in order to construct the initial RMP$_0^0$. In addition, the solution of RMP$_0^0$ must be feasible in order to be able to call the column generation algorithm. In this section, we describe the implementations used to initialize C_G.

Artificial variables

We introduce artificial variables to (ASPP-LRSP) in order to guarantee an initially feasible RMP_0^0. We create an imaginary facility (refer to as artificial facility) with $|I|$ pairings, one for each customer. Each pairing (column) visits one customer. Let t_a be the decision variable representing the artificial facility and be equal to 1 if it is open, and 0 otherwise. Let $z_{a_1}, .., z_{a_{|I|}}$ be the pairing variables associated with the artificial facility. z_{a_k} represents the pairing visiting customer k. We modify (ASPP-LRSP) by adding these artificial variables:

1. Add $C_a^F t_a + \sum_{i \in I} C_a z_{a_i}$ to the objective function (3.1), where C_a^F and C_a are costs of the artificial variables.

2. Modify constraints (3.2) by adding the artificial pairing variables:

$$\sum_{j \in J} \sum_{p \in P_j} a_{ip} z_p + z_{a_i} \; = \; 1 \quad \forall i \in I. \tag{3.17}$$

3. Add the following constraints related with new variables:

$$z_{a_i} - t_a \; \leq \; 0 \quad \forall i \in I \tag{3.18}$$

$$0 \leq z_{a_i} \; \leq \; 1 \quad \forall i \in I, \tag{3.19}$$

$$0 \leq t_a \; \leq \; 1. \tag{3.20}$$

Note that we do not actually need to add a variable for the artificial facility or to add constraints (3.18); we do that for purposes of interpretation and consistency with the model variables. We assign large costs to the artificial variables to discourage them being nonzero in the solution, unless it is infeasible. We let $C_a^F = (R+1)\text{max}_{j \in J}\{C_j^F\}$ and $C_a = 2C^O \sum_{i \in I} T_{1,i} + C^V$.

The model with artificial variables results in a feasible RMP_0^0. In addition, these artificial variables eliminate some of the cases in which the addition of the branching rules results in an initially infeasible RMP_0^i (without artificial variables) at any node i.

Initial Heuristic

From previous computational experience with column generation algorithms, we expect a reduction in the number of pricing iterations required to solve the LP relaxation when starting from a good set of columns rather than an easily constructed or arbitrary solution. To be able to improve the performance of the first column generation algorithm in our branch-and-price algorithm, we developed a heuristic algorithm in order to generate a set of columns and a feasible solution for the LRSP.

In order to construct feasible solutions for the integrated LRSP, we combine a facility location heuristic, several well-known VRP heuristics (Clark and Wright, nearest neighbor, nearest insertion and sweep heuristics), and a bin-packing heuristic (best fit decreasing). We construct pairings for a given subset of facilities by determining the following decisions sequentially:

- Allocation of customers to facilities,
- design of feasible vehicle routes, and
- assignment of routes to vehicles.

The total cost of such a sequentially constructed solution for a given subset of facilities is calculated considering the routing costs, vehicle and facility fixed costs.

To decide a subset of open facilities that would yield such a solution of lowest cost, the heuristic algorithm starts with all facilities open and iteratively closes the facility whose elimination results in the greatest cost savings until the problem becomes infeasible (this is called the DROP heuristic in the context of facility location problems).

We refer to the overall heuristic algorithm as I-Heur. I-Heur produces a set of columns as well as a feasible solution for the LRSP. I-Heur is composed of two main functions denoted by DROP(J) and TCOST(J). DROP(J) returns a subset of facility set J that results in the minimum cost calculated using function TCOST. TCOST(J_s) returns cost of a feasible solution to an LRSP instance in which all facilities in set J_s are open. TCOST(J_s), first allocates the customers to open

facilities in set J_s, then constructs vehicle routes for each facility to cover the customers assigned to each facility, and assigns routes to vehicles.

In order to present an outline of I-Heur, we define some notation. Let CW(j, I_j) denotes the application of the Clark and Wright algorithm for facility j and customers in set I_j. We use notation R = CW(j, I_j) to denote the set of vehicle routes generated at the end of the algorithm. Similarly, we use notation NN(j, I_j), NI(j, I_j), SW(j, I_j) to represent the applications of nearest neighbor, nearest insertion and sweep heuristics, respectively. Let (P, C) = BFD(R) denotes the application of the best fit decreasing heuristic to the routes in set R, which returns the set of pairings P and the associated cost (C) including routing and vehicle fixed cost. The outline of the algorithm I-Heur is presented in Figure 3.3.

I-Heur returns a set of columns for facilities that are selected to be open. We add these columns to RMP_0^0 and initialize the z_{UB} be the cost of the solution obtained from I-Heur. To balance the number of columns added for each facility, for the facilities that are fixed to be closed in the I-Heur solution, we determine a set of customers that can be served from the facility within the time limit and use function CW and BFD to determine a set of pairings for those facilities. We update column set C_G with these columns.

3.4.2 Overview

We implemented the algorithm in MINTO 3.1 using CPLEX 9.1 as the LP solver. MINTO (Savelsbergh and Nemhauser (1996)) is a solver for mixed integer linear programs that uses a branch-and-bound algorithm with an LP solver. MINTO has built-in applications such as preprocessing, constraint generation and primal heuristics, but it also allows the user to design his/her own application code such as a branch-and-cut or a branch-and-price that is specific to a problem. Within MINTO, we customized several routines and added new ones for our algorithm. We have written routines such as to generate an initial set of columns, do column generation and to select a branching variable. In the following, we explain some parts of the algorithm in detail.

I-Heur
Input: An instance of the LRSP defined by parameters and sets in Section 2.2.1.
Output: An upper bound for the optimal solution to the instance and a set of pairings.

DROP(J):

1. $J_o \leftarrow J$ (set of open facilities).
2. For all $k \in J_o$:
   i. Close facility k. $J^k \leftarrow J_o \backslash \{k\}$.
   ii. Call TCOST(J^k).
3. $k^* \leftarrow \text{argmin}_{k \in J_o}\{\text{TCOST}(J^k)\}$.
4. If TCOST(J^{k^*})$< \infty$, then $J_o \leftarrow J_o \backslash \{k^*\}$ and return to step 2.
5. Return set of open facilities, J_o, total cost TCOST(J_o) and set of pairings constructed in function TCOST(J_o).

TCOST(J_t):

1. Allocate customers, I, to facilities in set J_t. Assign each customer to the nearest facility whose capacity is not exceeded.
   Let I_j be the partition of I for all $j \in J_t$.
2. COST $\leftarrow \sum_{j \in J_t} C^F$ (fixed cost of open facilities).
3. For all $j \in J_t$:
   (a) $R_1 \leftarrow$ CW(j, I_j) and $(P_1, C_1) \leftarrow$ BFD(R_1).
   (b) $R_2 \leftarrow$ NN(j, I_j) and $(P_2, C_2) \leftarrow$ BFD(R_2).
   (c) $R_3 \leftarrow$ NI(j, I_j) and $(P_3, C_3) \leftarrow$ BFD(R_3).
   (d) $R_4 \leftarrow$ SW(j, I_j) and $(P_4, C_4) \leftarrow$ BFD(R_4).
   (e) COST $\leftarrow$ COST + min$\{C_1, C_2, C_3, C_4\}$ and $P_j \leftarrow P_{\text{argmin}\{C_1, C_2, C_3, C_4\}}$.
4. Return COST, and set of pairings P_j for all $j \in J_t$.

Figure 3.3: Outline of the Initial Heuristic I-Heur

Column Generation. In our column generation algorithm, we prefer the exact pricing algorithm 2p-ESPPRC to 1p-ESPPRC (described in Section 3.2.2). Our computational experiments in Section 5.4.1 show that 2p-ESPPRC is more efficient than 1p-ESPPRC. Therefore, in our branch-and-price algorithm, we use 2p-ESPPRC as an exact solution approach for the pricing problem.

In addition, computational experiments in Section 5.4.1 suggest that adding multiple columns

instead of a single best column at each call to 2p-ESPPRC performs better. Among the values we tested, addition of at most 40 columns (per facility) that have the smallest negative reduced cost performed better than the others. Therefore, in our implementation, at each iteration of the column generation algorithm, we generate at most 40 columns. Recall that pricing problem for each facility is solved independently. Therefore, in our column generation algorithm, the pricing algorithms are called sequentially for each open facility.

As discussed in Section 3.2.3, solving the pricing problem exactly by generating and processing all the labels can be time-consuming. To reduce the time spent, we use the two previously described heuristic versions of the pricing algorithm before employing the exact pricing algorithm. The benefit of applying heuristic pricing algorithms before 2p-ESPPRC is demonstrated computationally in Section 5.4.2. As in exact pricing algorithm, we add at most 40 columns at each call to the heuristic pricing algorithms.

To generalize the process, we present the steps of the overall column generation algorithm which we refer to as E-COLGEN(u, U, l, L), where u, U, l and L are input parameters associated with heuristic pricing algorithms in order to define their usage in the column generation algorithm.

Recall that in the heuristic pricing algorithm 2p-ESPPRC-CS, we need to set the subset size. We use notation P =2p-ESPPRC-CS(u) to represent a call to 2p-ESPPRC-CS with subset size u, where P is the set of pairings generated. Similarly, let P =2p-ESPPRC-LL(l) denote a call to 2p-ESPPRC-LL with a label limit l. Based on the given notation, we generalize the steps of the E-COLGEN in our branch-and-price algorithm in Figure 3.4.

In addition, as we mentioned before, in E-COLGEN each of the pricing algorithms 2p-ESPPRC-CS, 2p-ESPPRC-LL and 2p-ESPPRC is called sequentially for each open facility. Steps 2 to 5 are executed independently for each open facility, thus the label limit for each facility at any iteration may be different. For a facility, if 2p-ESPPRC does not return any columns, then we skip generating columns for that facility until $P = \emptyset$ for all non skipped facilities. Then, 2p-ESPPRC is called for all skipped facilities. If new columns are generated for any of them, we keep calling 2p-ESPPRC until $P = \emptyset$ for all facilities.

E-COLGEN(u, U, l, L)
Input: An instance P^i with column set C_G and parameters u, U, l and L.
Output: z^i_{LP} and column set C_G updated with new columns.

1. Do $P \leftarrow$2p-ESPPRC-CS(u) until $P = \emptyset$ or number of iterations $\geq U$.
2. If $l \leq L$, do $P \leftarrow$2p-ESPPRC-LL(l) until $P = \emptyset$.
   Otherwise, go to step 6.
3. Let $l = 2l$.
4. Execute $P \leftarrow$2p-ESPPRC.
5. If $P = \emptyset$, then $l = L + 1$ and STOP. Otherwise, return to Step 2.
6. Execute $P \leftarrow$2p-ESPPRC until $P = \emptyset$.

Figure 3.4: Column Generation Algorithm for 1S-EBP and 2S-EBP

Branch Selection. Given a fractional solution at a node of the branch-and-price tree, we apply the branching rules in the order that they are described in Section 3.3.1. Our computational experiments showed that the order of the application of the branching rules affects the performance of the algorithm significantly. For each branching rule, we choose the most fractional disjunction in each branching rule. In our computational experiments, choosing the most fractional disjunction in rule 3 and 4 results in slightly better than choosing the first disjunction that is found.

Primal Heuristic. Having good primal feasible solutions can reduce the size of the explored tree. We use I-Heur, MINTO's built-in primal heuristic routines, and a standard branch-and-bound algorithm to update the upper bound in the algorithm. As we mentioned earlier, we initialize the branch-and-price algorithm with the upper bound obtained from I-Heur. MINTO's primal heuristic routines search for an upper bound automatically at each node of the tree. We may also apply a standard branch-and-bound algorithm to problem P^0 with the current set of columns. This will provide an upper bound for P^0. We use this method only at the root node and subject to a one-hour time limit. In the next section, we will present another approach to find a primal feasible solution. We run a heuristic version of a branch-and-price algorithm (that includes only heuristic pricing

algorithms) which is a more powerful procedure to improve the upper bound. In Section 5.4.3, we present some computational experiments comparing the performance of the heuristic approaches used and the tightness of the upper bounds obtained in each.

3.4.3 Variants

We designed two versions of the branch-and-price algorithm for the LRSP: one-stage and two-stage branch-and-price algorithm. Figure 3.6 summarizes these two variants of the algorithm. The two-stage branch-and-price algorithm is an extended version of the one-stage algorithm, and it is designed based on the idea of "restart". The two-stage branch-and-price algorithm aims to solve larger instances. In this section, we present details of these two versions of the solution algorithm for the LRSP. In Section 5.4.4, we present computational experiments comparing the performance of the one-stage and the two-stage branch-and-price algorithms.

One-Stage Branch-and-Price Algorithm (1S-EBP). For small or medium size instances, we use a one-stage solution algorithm that is an exact branch-and-price algorithm, i.e., if it is run without a time limit, the algorithm terminates with z^*_{LRSP}. We refer to this version of the solution algorithm as 1S-EBP. 1S-EBP is initialized by using the upper bound and columns that are generated by I-Heur and employs E-COLGEN at each node of the branch-and-price tree.

Two-Stage Branch-and-Price Algorithm (2S-EBP). For larger instances, we designed a *two-stage* solution algorithm which is obtained by augmenting the one-stage branch-and-price algorithm with an improved version of initialization. We refer to this algorithm as 2S-EBP. To initialize the two-stage branch-and-price algorithm, we designed a heuristic version of a branch-and-price algorithm (which we refer to as HBP) which provides a good upper bound and a good set of columns. We describe HBP next in the section. The first stage of the two-stage branch-and-price algorithm employs HBP. Then, using the upper bound and the columns that are obtained from

H-COLGEN(u, U, l, L)
Input: An instance P^i with column set C_G and parameters u, U, l and L.
Output: An upper bound for z^i_{LP} and column set C_G updated with new columns.

1. Do $P \leftarrow$2p-ESPPRC-CS(u) until $P = \emptyset$ or number of iterations $\geq$ U.
2. Do $P \leftarrow$2p-ESPPRC-LL(l) until $P = \emptyset$.
3. Let $l = 2l$.
4. If $l \leq L$, go to step 2. Otherwise, STOP.

Figure 3.5: Column Generation Algorithm for HBP

HBP, an exact branch-and-price algorithm employing E-COLGEN at each node of the tree is executed as a second stage of the 2S-EBP. We can define 2S-EBP as a combination of HBP and 1S-EBP.

Heuristic Branch-and-Price Algorithm (HBP). In HBP, we construct an initial branch-and-price tree subject to a time limit, employing only the heuristic pricing algorithms 2p-ESPPRC-LL and 2p-ESPPRC-CS. The algorithm is initialized with I-Heur. When the time limit is reached (or the algorithm terminates), we have a set of generated columns that are available to be used and an upper bound for the problem. Using the notation that is given in the previous section, we present the main steps of the column generation algorithm (referred to as H-COLGEN) that is employed in HBP in Figure 3.5.

In H-COLGEN, we use smaller U, and smaller l and L compared to E-COLGEN. Small parameter values shorten the total execution time of H-COLGEN. HBP generates very good upper bounds for the LRSP instances. In Section 5.4.3, we present computational results evaluating the upper bounds obtained in HBP compared to other heuristic solution algorithms, and discuss the strength of the bounds with respect to z^0_{LP} and z^*_{LRSP}.

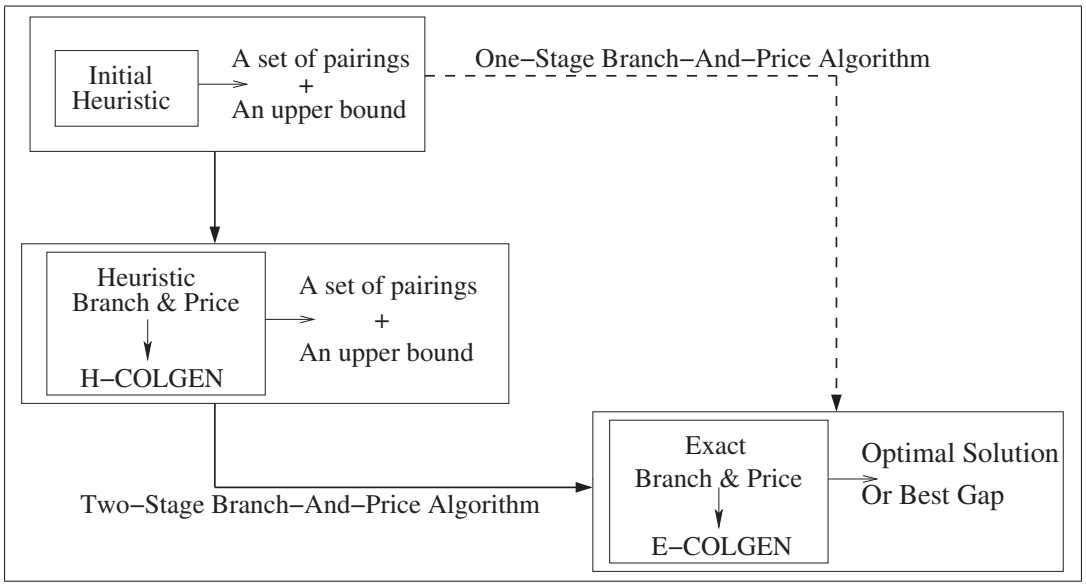


Figure 3.6: Variants of the Branch-and-Price Algorithm

Chapter 4

Valid Inequalities

In this chapter, we investigate classes of inequalities valid for the feasible region of the LRSP whose generation can be incorporated into our branch-and-price algorithm. Extending a branch-and-price algorithm to include methods for dynamically generating valid inequalities is a promising approach in the effort to improve the overall performance of the solution algorithm. The addition of cutting planes to the LP relaxation can improve bounds, but this comes at the expense of the additional time required to generate them and to solve the resulting larger LP relaxations, as well as the possible performance reduction in the column generation procedure caused by the incorporation of the cuts. Hence, efficiency of an extended algorithm depends on factors related to the branch-and-price algorithm other than the strength of the new valid inequalities.

The goal of this chapter is to provide an overview of related classes of valid inequalities, describe how to incorporate their generation into our branch-and-price algorithm and investigate the performance of the overall algorithm. In order to find cuts that are valid for the LRSP, we explore classes of valid inequalities derived from known relaxations of the LRSP, such as the knapsack problem, the set partitioning problem, the LRP and the VRP.

The main contributions of this chapter are to investigate the most common cuts valid for polytopes related to the LRSP polytope; discuss their validity for the LRSP; describe their integration into the (SPP-LRSP) model; describe the necessary updates to our branch-and-price algorithm;

and assess the value of these cuts to the solution algorithm. In addition, we explore possible ways of adapting the disjunctive cut procedure, commonly used in branch-and-cut algorithms. We are not aware of any publication that discusses the use of general disjunctive cuts in a branch-and-price algorithm. This chapter also provides some theoretical results about the strength of (SPP-LRSP) by showing that some of the cuts that can be derived using the graph-based formulation are necessarily satisfied by solutions to the LP relaxation of (SPP-LRSP).

4.1 Notation and Overview

Recall the steps of the algorithm referred to as BRANCH-CUT-PRICE presented in Figure 1.6. We use the same notation presented in Chapter 1. Since we want to extend the branch-and-price algorithm developed based on formulation (ASPP-LRSP) to a branch-cut-and-price algorithm, P^0 in the algorithm corresponds to (ASPP-LRSP). Therefore, we may use S^0 to denote the set of feasible solutions to problem (ASPP-LRSP). For ease of notation, we refer to the overall procedure that generates valid inequalities based on a given solution x^i_k as CUTGEN(x^i_k). This procedure returns a set of valid inequalities separating the solution x^i_k from the polyhedron defined by the constraints of RMP^i_k. Note that, in a generic BRANCH-CUT-PRICE, the cut generation procedure can be called at any time during the column generation algorithm. In the solution procedure for the LRSP, we only allow cut generation at the end of a complete column generation phase. Therefore, at Step 3(b) of BRANCH-CUT-PRICE, at iteration k of the column generation algorithm, if $C^i_k = \emptyset$ and if we want to search for valid inequalities, we proceed to Step 4 and call CUTGEN(x^i_k).

Recall that the pricing problem in the column generation algorithm is defined with respect to a particular LP relaxation of the problem and a particular dual solution to the current RMP. Therefore, whenever we add to P^i the set of valid inequalities generated, the pricing algorithm needs to be modified based on the classes of valid inequalities added to the subproblem. We do not express this modification explicitly in the steps of BRANCH-CUT-PRICE. However, in our BRANCH-CUT-PRICE algorithm, we only generate the valid inequalities for which we can

efficiently modify the solution algorithm for the pricing problem. Thus, it is important to discuss the characteristics of classes of valid inequalities that we can easily use in BRANCH-CUT-PRICE by modifying the pricing algorithm and the factors affecting the efficiency of the modified pricing algorithm. As we describe the valid inequalities for the LRSP in the remainder of this chapter, we describe particular modifications required for the pricing algorithm and point out which of the general categories detailed below each class falls into.

Classification of Valid Inequalities. Consider an inequality (α, β) valid for subproblem P^i of the form

$$\sum_{j \in J} \alpha_j t_j + \sum_{p \in P} \alpha_p z_p \geq \beta \quad \forall (t, z) \in S^i. \tag{4.1}$$

Let $\bar{P}$ be the set of columns generated so far and $(\bar{t}, \bar{z}) \in \mathbb{R}_+^{|J|+|P|}$ be an optimal fractional solution to the LP relaxation of P^i. Since we use only a restricted version of the LP relaxation of P^i in order to obtain $(\bar{t}, \bar{z})$, $\bar{z}_p = 0$ for all $p \in P \backslash \bar{P}$. In the cut generation procedure, while we generate an inequality (α, β) that is violated by $(\bar{t}, \bar{z})$, in order to guarantee that the inequality is valid for S^i, we need to consider and be able to calculate the coefficients of the columns in set $P \backslash \bar{P}$ in the inequality. We may assume that $\alpha_p = 0$ for all $p \in P \backslash \bar{P}$ while we generate (α, β), however, we need to be able to modify the coefficients α, β as we generate new columns.

To consider the non zero α coefficients for the columns to be generated, we need to modify the pricing problem and the pricing algorithm. In order not to reduce the overall efficiency of the branch-cut-and-price algorithm, we are interested in the question of for which type of valid inequalities we can modify the pricing algorithm without increasing its complexity significantly. Spoorendonk and Desaulniers (2008a) list two properties for a valid inequality that make it favorable in a column generation algorithm: (i) we can determine α_p for column $p \in P \backslash \bar{P}$ in a reasonable time, and (ii) we can easily modify the pricing algorithm in order to consider the dual variable associated with the inequality in the cost function of the pricing problem.

We classify the inequalities of the form (4.1) based on the structure of α to identify the cases in which we can easily incorporate a class of valid inequalities into our branch-cut-and-price algorithm. For a given solution, we let γ be the dual variable corresponding to the new inequality. We define three groups:

Group 1 For all $p \in P \backslash \bar{P}$, $\alpha_p = \sum_{(c,k)\in A} l_{ckp}\alpha'_{ck}$, where A is the set of arcs in the problem, $l_{ckp} = 1$ if pairing p visits arc (c,k) and α'_{ck} is fixed and known for all $(c,k) \in A$.

By modifying the cost of arc (c,k) by adding $\alpha'_{ck}\gamma$, we can easily handle the additional dual variable in the reduced cost of z_p. None of the pricing algorithms we use changes (conditions we check, domination rules, etc.) because we know the exact reduced cost of any partial path generated during the algorithm and how it will compare to other paths in future extensions. The inequalities that satisfy this condition can be used in our branch-cut-and-price algorithm without any significant changes in the pricing algorithm.

Group 2 For all $p \in P \backslash \bar{P}$, $\alpha_p = K$, where K is a known constant.

The inequalities satisfying this condition can easily be used in our branch-cut-and-price algorithm, since the reduced cost of any column is modified by a fixed amount $K\gamma$. This modification can be done after the pricing algorithm is terminated.

Group 3 An inequality that does not satisfy the conditions of Groups 1 and 2 is a Group 3 inequality. As in Groups 1 and 2, the reduced cost of variable z_p needs to be modified by adding $\alpha_p\gamma$. However, the value of α_p for all $p \in P \backslash \bar{P}$ may depend on the satisfaction of some conditions and may be different for any pairing p. In the pricing algorithm, consider two partial paths, p_1 and p_2. Let α_{p_1} and α_{p_2} be the coefficients, and $\hat{c}_{p_1}$ and $\hat{c}_{p_2}$ be the reduced costs calculated for each, respectively. We can compare $\hat{c}_{p_1}$ and $\hat{c}_{p_2}$ at the current iteration, but we do not know how these values will change in comparison to each other for extensions of these paths in future iterations, since we do not know what the coefficients of future extensions will be. Therefore, we cannot use the same domination rules that we are currently using to decide whether to keep or delete partial paths in any iteration of the algorithm. We

need to modify the domination rules in order to handle additional conditions that need to be considered.

Note that even if it is valid to set $\alpha_p = 0$ for all $p \in P \backslash \bar{P}$, the pricing problem still needs to be modified. Let q such that $\alpha_q > 0$ be one of the columns in C_G. Unless, we forbid generation of a pairing that is in the same symmetry group (pairings including the same set of routes) with pairing q, the column generation would generate a new column and replace z_q with it. In that case, the effect of the inequality on the LP relaxation bound is eliminated. It is usually very difficult (except in very simple cases) to forbid generation of a particular pairing.

Furthermore, for each class of inequalities (depending on the α) new domination rules need to be designed. We do not have an efficient way of handling any class of inequalities that can be counted in this group in the branch-cut-and-price algorithm. However, there are some studies in the literature (will be discussed later) using some classes of inequalities in this group with column generation algorithm.

Next, we investigate classes of valid inequalities for the LRSP. We discuss whether we can use them in BRANCH-PRICE-CUT and describe procedures to be embedded in CUTGEN procedures for the classes that can be included in BRANCH-PRICE-CUT.

4.2 Valid Inequalities For Set Partitioning-Based Formulations

In this section, we discuss some valid inequalities derived based on the set partitioning-based formulation of the LRSP (SPP-LRSP). In Chapter 2, we tightened this formulation with two sets of valid inequalities, (2.18) and (2.25), that are shown to be effective computationally in Section 5.3. Here, we study the LRSP polytope, defined as

$$\mathcal{P}_{\text{SPP}} = \text{conv}\{t \in \{0,1\}^{|J|}, z \in \{0,1\}^{\sum_{j \in J} |P_j|} | t, z \text{ satisfies (2.14), (2.15)}\}. \tag{4.2}$$

This study is two-fold: first, we investigate the valid inequalities for relaxations of polytope $\mathcal{P}_{\text{SPP}}$ and second, we investigate valid inequalities for reformulations of $\mathcal{P}_{\text{SPP}}$.

4.2.1 Relaxations

In this section, we investigate two relaxations: the knapsack problem and the set partitioning problem. We list the most common classes of valid inequalities for these problems (from the literature) based on our notation for (SPP-LRSP). Although we do not generate any of these classes of valid inequalities in our branch-cut-and-price algorithm (because they are group 3 inequalities), we provide this review of the literature and discuss possible implementation difficulties for completeness.

Knapsack Polytope. In $\mathcal{P}_{\text{SPP}}$, if we relax constraints (2.15) by dropping the variable t, then each constraint is a 0/1 knapsack constraint. Since we assume that the column sets for each facility, i.e., the items available for each knapsack, are disjoint from each other, these constraints decompose into $|J|$ single knapsack polytopes rather than one multiple knapsack polytope. We let $\mathcal{P}^j_{\text{KNP}}$ be the knapsack polytope that corresponds to facility $j \in J$, so that

$$\mathcal{P}^j_{\text{KNP}} = \text{conv}\{z \in \{0,1\}^{|P_j|} | \sum_{p \in P_j} D(p) z_p \leq L^F_j\} \quad \forall j \in J, \tag{4.3}$$

where $D(p) = \sum_{i \in I} a_{ip} D_i$, the total demand of customers in pairing p. For each $j \in J$, $\mathcal{P}^j_{\text{KNP}}$ can be seen as a relaxation of $\mathcal{P}_{\text{SPP}}$. Thus, we can investigate inequalities valid for $\mathcal{P}^j_{\text{KNP}}$, as these are also valid for (SPP-LRSP). Balas (1975), Padberg (1980), Crowder et al. (1983) and Weismantel (1997) are some of the authors who have studied the polyhedral structure of knapsack problems. Here, we evaluate the applicability of the following valid inequalities that are investigated in those studies to our algorithm.

One set of well-known nontrivial inequalities that are valid for $\mathcal{P}^j_{\text{KNP}}$ are the *cover inequalities*. Based on our notation and definition of $\mathcal{P}^j_{\text{KNP}}$, a set $R_j \subseteq P_j$ is called a cover if $\sum_{p \in R_j} D(p)$

$> L_j^F$. If the set R_j is minimal, then the following inequality is valid for $\mathcal{P}^j_{\text{KNP}}$:

$$\sum_{p \in R_j} z_p \;\leq\; |R_j| - 1 \quad \forall z \in \mathcal{P}^j_{\text{KNP}}. \tag{4.4}$$

Another set of nontrivial inequalities defined for $\mathcal{P}^j_{\text{KNP}}$ is the *(1-k)-configuration inequalities*. Let $R_j \subseteq P_j$ and $\sum_{p \in R_j} D(p) \leq L_j^F$. If $R' \cup \{q\}$ is a minimal cover for all $R' \subseteq R_j$ and $q \in P_j \backslash R_j$, then, the following inequality is valid for $\mathcal{P}^j_{\text{KNP}}$:

$$(r - k + 1) z_q + \sum_{p \in R_j^r} z_p \;\leq\; r \quad \forall z \in \mathcal{P}^j_{\text{KNP}}, \tag{4.5}$$

where r is some integer satisfying $k \leq r \leq |R_j|$, and R_j^r is a subset of R_j such that $|R_j^r| = r$. When $k = |R_j|$, the inequality (4.5) is a cover inequality; therefore, cover inequalities are also (1-k)-configuration inequalities.

The inequalities (4.4) and (4.5) are group 3 inequalities. At this time, it is not clear how to update pricing algorithms efficiently in order to handle these, so we did not utilize these in our branch-cut-and-price algorithm. If an efficient way of handling inequalities in the form of (4.4) and (4.5) could be found, we conjecture they would improve the performance of our algorithm as they have been very successful in the solution of $\mathcal{P}^j_{\text{KNP}}$.

Set Partitioning Polytope. Constraints (2.14) and the integrality constraints in $\mathcal{P}_{\text{SPP}}$ define a set partitioning polytope as follows:

$$\mathcal{P}_{\text{SP}} \;=\; \text{conv}\{z \in \{0,1\}^{\sum_{j \in J} |P_j|} | \sum_{j \in J} \sum_{p \in P_j} a_{ip} z_p = 1 \quad \forall i \in I\}. \tag{4.6}$$

Because $\mathcal{P}_{\text{SP}}$ is the intersection of set packing and set covering polytopes, valid inequalities for the set partitioning problem can be derived by investigating the polyhedral structure of these two polytopes (see Balas and Padberg (1976) for a review of polyhedral structures of all these polytopes). A common class of valid inequalities used in the literature is the *clique inequalities*. Let

$P = \cup_{j \in J} P_j$ and $I_p = \{i \in I | a_{ip} = 1\}$ for $p \in P$. For $R \subseteq P$ such that $I_{p_1} \cap I_{p_2} \neq \emptyset$ for all p_1, $p_2 \in R$, the following inequality is called a clique inequality:

$$\sum_{p \in R} z_p \leq 1 \quad \forall z \in \mathcal{P}_{\text{SP}}. \tag{4.7}$$

Inequality (4.7) enforces that only one pairing from such a set R may be selected. In order to generate violated inequalities of this form, the notion of a *clique* is used. A *clique* is a complete subgraph (a graph in which every pair of nodes are connected by an edge) of a given graph. In order to derive clique inequalities for $\mathcal{P}_{\text{SP}}$ based on this definition, a graph (generally called the *intersection graph* or the *conflict graph*) including one node for each variable of $\mathcal{P}_{\text{SP}}$ can be constructed. In this graph, there is an edge between a pair of nodes p_1 and p_2 if $I_{p_1} \cap I_{p_1} \neq \emptyset$ for all p_1, $p_2 \in P$. A valid inequality in the form of (4.7) can be written for each set of variables corresponding to a clique in this graph.

Another common class of valid inequalities used for $\mathcal{P}_{\text{SP}}$ is the *odd-hole inequalities* that are constructed from chordless cycles with an odd number of nodes in an intersection graph associated with an instance of the set partitioning problem. If $C \subseteq P$ is the set of variables representing nodes in an odd cycle, then the following is valid for $\mathcal{P}_{\text{SP}}$:

$$\sum_{p \in C} z_p \leq \left\lfloor \frac{|C|}{2} \right\rfloor \quad \forall z \in \mathcal{P}_{\text{SP}}. \tag{4.8}$$

There are other types of inequalities such as anti-hole, webs, and wheels (see Eso (1999) for details) that are derived from the intersection graphs in a similar way and have structure similar to (4.7) and (4.8). Here, our aim is to provide a short review of valid inequalities for the set partitioning problem. Since these inequalities are group 3 inequalities, we did not implement procedures for generating any of these cuts. We do not know of a way of modifying the pricing algorithm efficiently to incorporate these valid inequalities.

Spoorendonk and Desaulniers (2008b) developed a methodology in order to use clique inequalities in the column generation algorithm for the VRP. Jepsen et al. (2008) also developed a branch-cut-and-price algorithm for a VRP in which they provided a procedure to modify the pricing problem in order to use *subset-row inequalities* introduced based on the idea of clique and odd-hole inequalities. We review this work in order to further clarify the reasons that updating the pricing problem for inequalities in the form of (4.4) to (4.8) is not trivial and the structure of the cut is very important for a branch-cut-and-price algorithm. They constructed these inequalities based on information from the rows of the set partitioning polytope instead of considering a certain set of variables, as in clique and odd-hole inequalities. Based on our notation, Jepsen et al. (2008)'s subset-row inequalities are as follows

$$\sum_{j \in J} \sum_{p \in P_j} \Big\lfloor \frac{1}{k} \sum_{i \in R} a_{ip} \Big\rfloor z_p \leq \Big\lfloor \frac{|R|}{k} \Big\rfloor \quad \forall z \in \mathcal{P}_{\text{SP}}, \tag{4.9}$$

where $R \subseteq I$ (rows in set partitioning constraints) and $0 < k \leq |R|$. A column corresponding to a pairing visiting at least k of the customers in set R has a nonzero coefficient in inequality (4.9); otherwise, the coefficient is zero. They used this condition to modify the reduced cost calculation for a column in the pricing problem, which is also an ESPPRC.

In this case, the reduced cost of a column should be augmented by the term $-\gamma \Big\lfloor \frac{\sum_{i \in R} a_{ip}}{k} \Big\rfloor$ where γ is the dual variable that corresponds to constraint (4.9). To do that, they maintained a value $m_l = (|R \cap V_l| \bmod k)$ for each label l in the ESPPRC algorithm where V_l is the set of customers in label l. They then modified the domination rules based on the value of m_l for each label l. They showed that these inequalities were effective and improved the performance of their branch-cut-and-price algorithm. We continue our investigation by looking at the polytopes obtained by reformulating the (SPP-LRSP).

4.2.2 Reformulations

In order to overcome the difficulty of incorporation of the valid inequalities for the relaxations of $\mathcal{P}_{\text{SPP}}$ reviewed in the previous section into the pricing problem, we propose here a change of variables. We define $v_{ji} = \sum_{p \in P_j} a_{ip} z_p$ for all $j \in J$ and $i \in I$ and replace the variables z. The polytope $\mathcal{P}^r_{\text{SPP}}$ is then

$$\mathcal{P}^r_{\text{SPP}} = \text{conv } \{(t,v) \in \{0,1\}^{|J|+|J|\times|I|} | \sum_{j \in J} v_{ji} = 1 \quad \forall i \in I, \tag{4.10}$$

$$\sum_{i \in I} D_i v_{ji} - L^F_j t_j \leq 0 \quad \forall j \in J\}. \tag{4.11}$$

We can interpret v_{ji} as the fraction of the demand of customer $i \in I$ assigned to facility $j \in J$. Again, we investigate some relaxations of the polytope $\mathcal{P}^r_{\text{SPP}}$ to derive valid inequalities. The valid inequalities obtained can then be rewritten in terms of the original variables and used in our algorithm. We implemented two classes of the valid inequalities that will be presented next in this section: lifted cover inequalities and generalized assignment polytope inequalities. We implemented these in our branch-cut-and-price algorithm and presented some computational results in Section 5.5.1.

Multiple Knapsack Polytope. If we replace variable t_j with its upper bound 1 in inequalities (4.11) and relax the equality restriction in (4.10), the resulting inequalities along with the binary restriction for variable v define a multiple knapsack polytope:

$$\mathcal{P}_{\text{MKN}} = \text{conv } \{v \in \{0,1\}^{|J|\times|I|} | \sum_{j \in J} v_{ji} \leq 1 \quad \forall i \in I, \tag{4.12}$$

$$\sum_{i \in I} D_i v_{ji} \leq L^F_j \quad \forall j \in J\}. \tag{4.13}$$

Valid inequalities for the single knapsack polytope are also valid for $\mathcal{P}_{\text{MKN}}$. The cover inequalities and (1-k) configuration inequalities described in Section 4.2.1 can be applied here.

Let $R \subseteq I$ be a minimal cover for facility j ($\sum_{i \in R} D_i > L^F_j$ and $\sum_{i \in R \setminus \{k\}} D_i \leq L^F_j$ for all

$k \in R$). Then, we have

$$\sum_{i \in R} v_{ji} \leq |R| - 1, \tag{4.14}$$

valid for $\mathcal{P}_{\text{MKN}}$ and $\mathcal{P}^r_{\text{SPP}}$. The inequalities (4.14) can be strengthened using the lifting procedure described by Nemhauser and Wolsey (1989) and Gu et al. (1998), which we implemented in our branch-cut-and-price algorithm. MINTO has a built-in function to detect violated cover inequalities for 0-1 integer programs and lift them. This function implements the heuristic algorithm proposed by Gu et al. (1998) to search for the lifted cover inequalities of the form

$$\sum_{i \in R_1} v_{ji} + \sum_{i \in R_2} \gamma_{ji} v_{ji} + \sum_{i \in I \setminus R} \alpha_{ji} v_{ji} \leq |R_1| - 1 + \sum_{i \in R_2} \gamma_{ji} \quad \forall v \in \mathcal{P}_{\text{MKN}}, \tag{4.15}$$

where $R = (R_1 \cup R_2) \subseteq I$ is a minimal cover for j and $R_1 \neq \emptyset$. We employed the MINTO function in our algorithm to detect lifted cover inequalities violated by the portion of a given solution consisting of the variables associated with a single facility $j \in J$.

An inequality of the form (4.15) can rewritten in terms of the variables of $\mathcal{P}_{\text{SPP}}$ as follows:

$$\sum_{i \in R_1} \sum_{p \in P_j} a_{ip} z_p + \sum_{i \in R_2} \sum_{p \in P_j} \gamma_{ji} a_{ip} z_p + \sum_{i \in I \setminus R} \sum_{p \in P_j} \alpha_{ji} a_{ip} z_p \leq |R_1| - 1 + \sum_{i \in R_2} \gamma_{ji}, \tag{4.16}$$

for all $(R_1 \cup R_2) \subseteq I$, $j \in J$ and $\forall (t, z) \in \mathcal{P}_{\text{SPP}}$. An inequality of the form (4.16) is a group 1 inequality and can easily be added to our branch-cut-and-price algorithm without destroying the pricing problem structure. Let ω^j_R be the dual variable corresponding to inequality (4.16) associated with customer set R and facility j. To modify our algorithm to accommodate these inequalities, we modify the cost of incoming arcs for customer i in sets R_1, R_2 and $I \setminus R$ at facility j by ω^j_R, $\gamma_{ji}\omega^j_R$ and $\alpha_{ji}\omega^j_R$, respectively. These additional costs for single customers result in a modified total reduced cost for a column with a nonzero coefficient in inequality (4.16).

In our cut generation procedure, for a given fractional solution $(\bar{t}, \bar{z}) \in \mathbb{R}_+^{|J| + \sum_{j \in J} |\bar{P}_j|}$ to

RMP^i_k, where $\bar{P}_j$ is the current set of pairings for facility $j \in J$, we do the following.

- For each facility $j \in J$, if $\bar{t}_j = 1$, we find $\bar{v}^j \in \mathbb{R}^{|I|}_+$ using the transformation $\bar{v}_{jl} = \sum_{p \in \bar{P}_j} a_{lp} \bar{z}_p$ for all $l \in I$.
- For a given $\bar{v}^j$, we try to generate a violated lifted cover inequality in the form of (4.15) using MINTO's function.
- If the function returns a cut, then we rewrite it in the form of (4.16) and add it to F^i_k.

As mentioned earlier, computational results for the implementation of this set of valid inequalities are presented in Section 5.5.1. The (1-k) configuration inequalities are also valid for the single knapsack polytope; thus, it would also be possible to derive (1-k) configuration inequalities for $\mathcal{P}^r_{\text{SPP}}$ in terms of variable v. In a procedure similar to that for cover inequalities, (1-k) configuration inequalities in terms of variable v could be generated in a branch-cut-and-price algorithm without requiring significant changes in the pricing algorithms. Based on the computational experiments for the lifted cover inequalities, which are a special case of (1-k) configuration inequalities, however, we decided not to implement (1-k) configuration inequalities in our study.

Generalized Assignment Polytope. The generalized assignment problem (GAP) is a generalization of the multiple knapsack problem. The generalization is that the GAP may be formulated with different weights for each item (coefficients in (4.13)) for each knapsack. The generalized assignment polytope can be defined as follows:

$$\mathcal{P}_{\text{GAP}} = \text{conv}\{v \in \{0,1\}^{|J|\times|I|} | v \text{ satisfy (4.10) and } \sum_{i \in I} D_i v_{ji} \leq L^F_j \;\; \forall j \in J\}.$$

Valid inequalities for the knapsack polytopes are also valid for $\mathcal{P}_{\text{GAP}}$. Besides the inequalities for the knapsack polytope, there are only a few papers investigating the facets and valid inequalities of GAP. Some of these are Gottlieb and Rao (1990) and de Farias and Nemhauser (2001). Both papers relaxed the set partitioning constraints (4.10) and replaced them with set packing constraints.

In this relaxed GAP, de Farias and Nemhauser (2001) added the condition that for all $i \in I$, there is at most one $j \in J$ such that $v_{ji} > 0$. The following theorem from de Farias and Nemhauser (2001) presents a valid inequality for GAP.

Theorem 4.1 *[de Farias and Nemhauser (2001)] Let $R \subseteq I$ and $k \in J$. Suppose that $\sum_{i \in R} D_i > L_k^F$ and $\sum_{i \in R-\{r\}} d_i < L_k^F$ for some $r \in R$. Then,*

$$\sum_{i \in R} D_i v_{ki} + \sum_{i \in R} \left(max\{0, L_k^F - \sum_{a \in R-\{i\}} D_a\} \right) \sum_{j \in J-\{k\}} v_{ji} \leq L_k^F \quad \forall v \in \mathcal{P}_{GAP} \quad (4.17)$$

induces a facet of the relaxed GAP polytope.

The interpretation of (4.17) is simple: if the total demand of the customers in set R exceeds the capacity of facility k, then there must be a customer in R that is served by a facility other than k. If the second part of the left hand side of (4.17) is zero, it is equivalent to the facility capacity constraint for k, which is one of the trivial inequalities. However, in a non integer solution $(\bar{t}, \bar{v}) \in \mathbb{R}_+^{|J|+|J|\times|I|}$, if $\sum_{i \in I} D_i \bar{v}_{ki} = L_k^F$ and $\bar{v}_{ki} > 0$ for all $i \in R$, then there exists some $h \in J$ such that $\bar{v}_{hi} > 0$ for some $i \in R$. In this case, if $L_k^F - \sum_{a \in R-\{r\}} D_a > 0$ for some $r \in R$ satisfying $\bar{v}_{hr} > 0$, then the inequality (4.17) is violated. We can rewrite inequality (4.17) for $\mathcal{P}_{\text{SPP}}$:

$$\sum_{i \in R} D_i \sum_{p \in P_k} a_{ip} z_p + \sum_{i \in R} \left(\max\{0, L_k^F - \sum_{a \in R-\{i\}} D_a\} \right) \sum_{j \in J-\{k\}} \sum_{p \in P_j} a_{ip} z_p \leq L_k^F, \quad (4.18)$$

where $\sum_{i \in R} D_i > L_k^F$. Since the inequalities of the form (4.18) are in group 1, they can be added to the algorithm without destroying the structure of the pricing problem.

We implemented this class of inequalities, which we refer to as GAP inequalities, and modified our branch-cut-and-price algorithm to include the generation of valid inequalities of the form of (4.18). We can incorporate GAP inequalities into our branch-cut-and-price algorithm by slightly modifying the cost function in the pricing problem. Let γ_R^k be the dual variable corresponding to

the inequality (4.18) for customer set R and facility k. Then we modify the cost of the incoming arcs of customers in R by adding either $D_i\gamma_R^k$ (for facility k) or $(\max\{0, L_k^F - \sum_{a\in R-\{i\}} D_a\})\gamma_R^k$ (for all facilities other than k).

de Farias and Nemhauser (2001) present a separation heuristic for this class of inequalities. Figure 4.1 describes the heuristic (referred to as GAP-Heur) in terms of our notation. We modified GAP-Heur slightly in our implementation by requiring in step 1(i) that the facility variable is equal to 1. To generate GAP inequalities violated by a given fractional solution $(\bar{t}, \bar{z}) \in \mathbb{R}_+^{|J|+\sum_{j\in J}|\bar{P}_j|}$, where $\bar{P}_j$ is the current set of pairings for facility $j \in J$, we do the following.

- We determine $(\bar{t}, \bar{v}) \in \mathbb{R}^{|J|+|J|\times|I|}$ using the transformation $\bar{v}_{ji} = \sum_{p\in\bar{P}_j} a_{ip}\bar{z}_p$ for all $i \in I$ and $j \in J$.
- We call GAP-Heur to separate cuts in the form of inequality (4.17).
- We rewrite the inequality in the form of (4.18).

In our experience with the LRSP instances, GAP-Heur fails to find violated cuts most of the time. The reason for this failure is condition (4.20) in Figure 4.1. Generally, condition (4.20) is not satisfied even though the facility capacity constraint is binding and condition (4.19) is satisfied. In the LRSP, we are dealing with the assignment of pairings to the facilities. Therefore, fractional pairing variables lead to multiple customers (rather than a single customer) that are served by more than one facility. In a fractional solution, for some facility, even though the capacity is fully utilized, it is not enough to remove the assignment of a single customer to this facility in order to force the total demand of the remaining customers to be less than the facility capacity since more than one customer is fractionally served by that facility. To increase the effectiveness of the procedure, we modified the method of determining the set R in the step 1 (ii) of the GAP-Heur, so that we may shrink this set in order to obtain a violated cut.

In our algorithm, we first execute steps 1 (ii) and 1 (iii) as in the original GAP-Heur procedure by setting $R = \{i \in I | v_{ki} > 0\}$. If we cannot find a violation, we reduce the size of set R by removing one customer at a time in the order of non increasing demand until a violation is found

GAP-Heur [de Farias and Nemhauser (2001)]
Input: An instance of the GAP and a fractional solution to its LP relaxation, $\bar{v} \in \mathbb{R}_+^{|J|\times|I|}$.
Output: A set of valid inequalities of the form (4.17) violated by $\bar{v}$.

1. For each $k \in J$:

   i. If $\sum_{i\in I} D_i \bar{v}_{ki} = L_k^F$, then proceed. Otherwise go to step 1 and pick the next facility.

   ii. Let $R = \{i \in I | \bar{v}_{ki} > 0\}$ and $D(R) = \sum_{i\in R} D_i$.
   If $D(R) > L_k^F$, then for each $c \in R$, check the following conditions:
   - Is c also served by another facility?

$$1 > \bar{v}_{kc} > 0, \text{ and } \sum_{j\in J\setminus\{k\}} \bar{v}_{jc} > 0 \tag{4.19}$$

   - When we remove c from the assignment set of facility k, does the full assignment of the rest of the customers to k become feasible?

$$D(R\setminus\{c\}) < L_k^F \tag{4.20}$$

   iii. If there is at least one $c \in R$ satisfying both (4.19) and (4.20), then (4.17) is violated for facility k and customer set R. Return a GAP cut for facility k.

Figure 4.1: Heuristic Separation Algorithm for GAP inequalities [de Farias and Nemhauser (2001)]

or $D(R) \leq L_k^F$. The steps of the updated algorithm are listed in Figure 4.2. As mentioned earlier, computational results for the implementation of this class of valid inequalities are presented in Section 5.5.1.

Capacitated Facility Location Polytope. If we relax the integrality of variable v in the constraints of $\mathcal{P}^r_{\text{SPP}}$, we get the capacitated facility location polytope:

$$\mathcal{P}_{\text{CFL}} = \text{conv } \{t \in \{0,1\}^{|J|}, v \in \mathbb{R}^{|J|\times|I|} | (t,v) \text{ satisfies } (4.10), (4.11), \text{ and } v_{ji} \in [0,1] \quad \forall j \in J, i \in I\}.$$

GAP-Heur-2
Input: An instance of the GAP and a fractional solution to its LP relaxation, $(\bar{t}, \bar{v}) \in \mathbb{R}_+^{|J|+|J|\times|I|}$.
Output: A set of valid inequalities of the form (4.18) violated by $(\bar{t}, \bar{v})$.

1. For each facility $k \in J$:
   i. If $\bar{t}_k = 1$ and $\sum_{i\in I} D_i \bar{v}_{ki} = L_k^F$, then proceed. Otherwise return to step 1.
   ii. Let $R = \{i \in I | \bar{v}_{ki} > 0\}$. If $D(R) > L_k^F$, then for each $c \in R$, check the conditions (4.19) and (4.20).
   iii. If there is at least one $c \in R$ satisfying (4.19) and (4.20), then return a cut for facility k and set R and return to step 1.
   iv. Sort the customers in R, $i_1 < i_2 < .. < i_{|R|}$ such that $D_{i_p} \geq D_{i_q}$ for all $p, q = 1, .., |R|$ and $i_p < i_q$.
   v. For $e = 1$ to $|R|$:
      a. Let $R = R - \{i_e\}$. If $D(R) \leq L_k^F$, then return to step 1.
      b. If there is at least one $c \in R$ satisfying (4.19) and (4.20), then return a cut for facility k and set R and return to step 1.

Figure 4.2: Outline of the Modified Heuristic Separation Algorithm for GAP inequalities for the LRSP

Therefore, if the valid inequalities for $\mathcal{P}_{\text{CFL}}$ are rewritten in terms of variables (t, z), they are valid for $\mathcal{P}_{\text{SPP}}$. Indeed, $\mathcal{P}^r_{\text{SPP}}$, the capacitated facility location polytope with binary restriction on variable v, is called the single source capacitated facility location polytope. We are not aware of any studies that investigate specifically the polyhedral structure of CFLPSS. Thus, we review the studies for $\mathcal{P}_{\text{CFL}}$, of which there are a few. Aardal et al. (1995), Leung and Magnanti (1989) and Padberg et al. (1985) investigated valid inequalities and gave conditions for these inequalities to be facet-inducing for $\mathcal{P}_{\text{CFL}}$. We did not implement generation procedures for any of these classes of valid inequalities, but here we give a brief list of common ones.

One class of valid inequalities is the *knapsack cover inequality*, defined in terms of the facility variables t. Aardal et al. (1995) call a subset M of the facilities a cover if $\sum_{j\in M} L_j^F > \sum_{j\in J} L_j^F - D(I)$. Based on this definition, they stated the following theorem:

Theorem 4.2 *[Aardal et al. (1995)] If M is a minimal cover with respect to J and I, $L^F_{min} = min_{j\in M}\{L^F_j\}$, and $\sum_{j\in J\setminus M} L^F_j + L^F_{min} > D(I)$, then the knapsack cover inequality*

$$\sum_{j\in M} t_j \geq 1 \quad \forall (t,v) \in \mathcal{P}_{CFL} \tag{4.21}$$

is facet-inducing for $\mathcal{P}_{CFL} \cap \{t \in \{0,1\}^{|J|} : t_j = 1 \ \forall j \in J\setminus M\}$.

This theorem requires that at least one facility from set M must be open because the total capacity of the facilities in the set $J\setminus M$ is not enough to cover the total demand of customers. Since (4.21) is valid for $\mathcal{P}_{\text{CFL}}$, then it is also valid for $\mathcal{P}^r_{\text{SPP}}$ and $\mathcal{P}_{\text{SPP}}$. Since the inequality defined in Theorem (4.2) only has nonzero coefficients for the t variables, we can add this inequality without affecting the structure of the pricing problem.

We did not implement the separation algorithms for this inequality in our study because of the effect of inequality (2.18) in (ASPP-LRSP). In the LRSP instances used for our computational results, the facility capacity values are all equal. Based on this, when all possible inequalities of the form (4.21) summed, we get $\sum_{j\in J} t_j \geq m$ where $m \in \mathbb{R}_+$. With some algebra, we can easily show that the minimum number of required facilities R (in (2.18)) is greater than m, if $1 < R < |J|$ (which is the case in our instances). Thus, the inequality (2.18) requiring that the total number of facilities must be greater than or equal to R is stronger than the inequality obtained by summing all possible inequalities in the form of (4.21). In addition, we do not add the knapsack cover inequalities to (ASPP-LRSP), since the addition of inequality (2.18) generally results in integer values for the facility variables in the LP relaxation solution to (ASPP-LRSP) and its subproblems in the tree.

Another class of valid inequality for $\mathcal{P}_{\text{CFL}}$ is the *flow cover inequalities*. Aardal et al. (1995) and Padberg et al. (1985) defined M as a flow cover if $M \subset J$ and $\sum_{j\in M} L^F_j > D(I)$. Aardal et al. (1995) and Padberg et al. (1985) presented the following theorem (expressed in terms of the notation in this section) that describes the flow cover inequalities:

Theorem 4.3 *[Aardal et al. (1995)] Let $M \subset J$ be a flow cover with respect to J and I and $\sum_{j\in M} L_j^F = D(I) + \lambda$. If max$_{j\in M}\{L_j^F\} > \lambda$ and $\sum_{j\in J} L_j^F > D(I) + L_r^F \ \forall r \in M$, then the flow cover inequality*

$$\sum_{j\in M}\sum_{i\in I} D_i v_{ji} + \sum_{j\in M}(L_j^F - \lambda)^+(1-t_j) \ \leq \ D(I) \quad \forall (t,v) \in \mathcal{P}_{CFL} \tag{4.22}$$

is facet-inducing for $\mathcal{P}_{CFL}$.

If all customers are assigned to the facilities in M, the excess capacity λ leads at least one facility in set M to be fractional in any solution to the LP relaxation associated with $\mathcal{P}_{\text{CFL}}$. The aim of the inequality (4.22) is to eliminate the effect of excess capacity. The second part in the left-hand side of the inequality forces all facilities in set M to take value close to 1. In inequality (4.22), if we replace the variable v_{ji} with $\sum_{p\in P_j} a_{ip} z_p$, we have the following corresponding inequality for $\mathcal{P}_{\text{SPP}}$:

$$\sum_{j\in M}\sum_{p\in P_j}\sum_{i\in I} D_i a_{ip} z_p + \sum_{j\in M}(L_j^F - \lambda)^+(1-t_j) \ \leq \ D(I) \quad \forall (t,z) \in \mathcal{P}_{\text{SPP}}. \tag{4.23}$$

In a procedure similar to that which we described before, we can incorporate the inequality (4.23) into a branch-cut-and-price algorithm since it is a group 1 inequality. However, we did not implement procedures for generating the class of this inequalities in our study because in most of our LP relaxation solutions $\bar{t} \in \mathbb{Z}_+^{|J|}$.

4.3 Valid Inequalities Based on Graph-Based Formulations

Another approach to deriving problem-specific classes of valid inequalities for the LRSP is to study the LRSP polytope $\mathcal{P}_{\text{GBP}}$ defined as the convex hull of the set of feasible solutions to (G-LRSP). When a valid inequality is derived in terms of the variables in (G-LRSP), then this inequality can be transformed to one valid for (SPP-LRSP) by transformation of variables using

the equalities (2.30)- (2.33) defined in the DWD of (G-LRSP) in Section 2.4.

The literature does not include any studies devoted to polyhedral structure of the LRSP. However, there are studies that investigate valid inequalities for the VRP and the LRP, each of which are relaxations of the LRSP. In this section, we investigate some sets of valid inequalities derived for the VRP or the LRP and originally defined in terms of graph-based variables. We discuss how we can incorporate the generation of these inequalities into our algorithm for solving (SPP-LRSP) and whether these inequalities have a positive impact. In fact, we either learned from the literature or proved herein that most of these inequalities are already satisfied by all solutions to the LP relaxation of (SPP-LRSP).

The most commonly used formulations in studies of the polyhedral structure of the VRP and the LRP are two-index vehicle flow formulations. These formulations use binary variables associated with the arcs between pairs of customer or facility locations as in Section 2.2.1. Valid inequalities for the two-index formulations can be transformed to one valid for (SPP-LRSP) by using the transformation

$$x_{ik} = \sum_{j \in J} \sum_{p \in P_j} l_{ikp} z_p \quad \forall (i,k) \in A, \tag{4.24}$$

where x_{ik} is the aforementioned binary variable and $l_{ikp} = 1$ if arc (i,k) is in pairing p. In some graph-based formulations, like (G-LRSP), variable x has also a third index associated with the vehicles in the problem. By summing the three-index variables associated with an arc for all vehicles in the problem, we can obtain the corresponding two-index variable associated with the same arc.

Certain classes of valid inequalities used in branch-and-cut algorithms for the VRP or the LRP are satisfied by all solutions to the LP relaxations of the set partitioning-based formulations of these problems that are analogous to our formulation. In the literature, there are a limited number of studies investigating the theoretical relationship between set partitioning-based formulations and graph-based formulations of the VRP, but there are none for the LRP. Letchford and

González (2006) proved theoretically that a solution to the LP relaxation of a set partitioning-based formulation for the VRP (with elementary resource constrained paths) implies some of the valid inequalities (such as *knapsack large multistar, subtour elimination, generalized large multistar, hypotour-like inequalities*) that are derived for a graph-based formulation in the two index space. This result is also applicable to the LRSP, which means that LP relaxation solutions to (SPP-LRSP) satisfy these type of cuts. Ropke and Cordeau (2009) did a similar study for the VRPTW with pickup and delivery, investigating the relationship between a branch-and-cut and branch-cut-and-price algorithm for that problem. They showed that some of the valid inequalities for the problem (e.g., *fork inequalities, reachability inequalities, precedence inequalities*) are implied by the set partitioning-based formulation of the problem obtained with a pricing problem that generates elementary resource constrained paths.

We started our study by investigating two of the most common classes of inequalities for the VRP: fractional capacity inequalities and rounded capacity inequalities.

Fractional Capacity Inequalities. Letchford and González (2006) noted that fractional capacity inequalities are dominated by generalized large multistar (GLM) inequalities and GLM inequalities are a subclass of the knapsack large multistar inequalities, which are proved to be implied by the set partitioning-based formulation of the VRP. Therefore, the results of Letchford and González (2006) prove that fractional capacity inequalities are redundant for the set partitioning formulation. Here, we re-derive this result. Let $\mathcal{P}_{\text{VRP}}$ be the VRP polytope based on the two-index vehicle flow formulation. Then the fractional capacity inequality for $R \subseteq I$ is as follows:

$$\sum_{i \in R} \sum_{k \in V \setminus R} x_{ik} \geq \frac{\sum_{i \in R} D_i}{L^V} \quad \forall x \in \mathcal{P}_{\text{VRP}}. \tag{4.25}$$

In other words, the total capacity of the vehicles leaving R must be greater than the total demand

of customers in R. Using equation (4.24), we can rewrite inequality (4.25) for the (SPP-LRSP) as

$$\sum_{j\in J}\sum_{p\in P_j}\sum_{i\in R}\sum_{k\in V\setminus R} l_{ikp}z_p = \sum_{j\in J}\sum_{p\in P_j}\alpha_p z_p \geq \frac{\sum_{i\in R} D_i}{L^V} \quad \forall (t,z)\in \mathcal{P}_{\text{SPP}}, \tag{4.26}$$

where α_p is the coefficient that corresponds to z_p for all $p \in \cup_{j\in J} P_j$. Here, α_p is the number of arcs from set R to set $V\setminus R$ visited in pairing p. As before, we let $I_p = \{i \in I | a_{ip} = 1\}$. For ease of notation, define n_K as a bound on the minimum number of vehicle routes that are needed to cover customer demand in set K, i.e., $n_K = \lceil \frac{\sum_{i\in K} D_i}{L^V} \rceil$. Since a pairing (say p) represents a feasible vehicle schedule, by construction, the total number of vehicles leaving customers in set $K \subseteq I_p$ is at least n_K. The following lemma derives from this fact.

Lemma 4.1 *For pairing p, let $K = I_p \cap R$, where R is the set of customers with respect to which an inequality of the form (4.25) is stated. If $K \neq \emptyset$, then the contribution (coefficient) of variable z_p to the left hand side of the inequality is at least n_K. In other words, $\alpha_p \geq n_K$.*

Let $(\bar{t}, \bar{z}) \in \mathbb{R}_+^{|J|+|\cup_{j\in J}\bar{P}_j|}$ be a solution to the LP relaxation of (SPP-LRSP) where $\bar{P}_j$ is the current set of pairings for facility $j \in J$.

Proposition 4.1 *If $(\bar{t}, \bar{z})$ satisfies (2.14) and (2.15), then it also satisfies (4.26).*

Proof Let us first multiply constraint (2.14) for each customer i by its demand D_i for all $i \in R$ and then sum over all customers in set R:

$$\begin{aligned}\sum_{j\in J}\sum_{p\in\bar{P}_j}\sum_{i\in R} D_i a_{ip}\bar{z}_p &= \sum_{i\in R} D_i,\\ \sum_{j\in J}\sum_{K\subseteq R}\sum_{p\in\bar{P}_j^K} D(K)\bar{z}_p &= D(R),\end{aligned}$$

where $\bar{P}_j^K \subseteq \bar{P}_j$ such that $I_p \cap R = K$ for all $p \in \bar{P}_j^K$, $K \subseteq R$ and $j \in J$. Note that $\bar{P}_j^{K_1} \cap \bar{P}_j^{K_2} = \emptyset$ for all $K_1, K_2 \subseteq R$ such that $K_1 \neq K_2$ and K_1, K_2 are nonempty. Since the pairings are feasible with respect to vehicle capacity by construction, $n_K L^V \geq D(K)$ for all

$K \subseteq R$. Therefore,

$$\sum_{j\in J}\sum_{K\subseteq R}\sum_{p\in \bar{P}_j^K} n_K L^V \bar{z}_p \geq \sum_{j\in J}\sum_{K\subseteq R}\sum_{p\in \bar{P}_j^K} D(K)\bar{z}_p \;=\; D(R),$$

$$\sum_{j\in J}\sum_{K\subseteq R}\sum_{p\in \bar{P}_j^K} n_K \bar{z}_p \;\geq\; \frac{D(R)}{L^V}.$$

From lemma 4.1:

$$\sum_{j\in J}\sum_{K\subseteq R}\sum_{p\in \bar{P}_j^K} \alpha_p \bar{z}_p \;\geq \sum_{j\in J}\sum_{K\subseteq R}\sum_{p\in \bar{P}_j^K} n_K \bar{z}_p \;\;\geq\;\; \frac{D(R)}{L^V}.$$

Therefore, inequality (4.26) is satisfied for any solution to the LP relaxation of the (SPP-LRSP). □

Rounded Capacity Inequalities. Another common class of inequalities is rounded capacity inequalities. These inequalities dominate the fractional capacity inequalities and they have the following form:

$$\sum_{i\in R}\sum_{k\in V\setminus R} x_{ik} \;\geq\; n_R \quad \forall x \in \mathcal{P}_{\text{VRP}}, \tag{4.27}$$

where $n_R = \lceil \frac{\sum_{i\in R} D_i}{L^V} \rceil$ for some customer set R. Here, we investigate whether the inequalities obtained by mapping these rounded capacity inequalities to the variable space of $\mathcal{P}_{\text{SPP}}$ can improve the LP relaxation associated with $\mathcal{P}_{\text{SPP}}$. Based on equation (4.24), if we rewrite (4.27) in terms of z variables, we obtain:

$$\sum_{j\in J}\sum_{p\in P_j}\sum_{i\in R}\sum_{k\in V\setminus R} l_{ikp} z_p = \sum_{j\in J}\sum_{p\in P_j} \alpha_p z_p \;\geq\; n_R \quad \forall (t,z) \in \mathcal{P}_{\text{SPP}}. \tag{4.28}$$

Let $(\bar{t}, \bar{z}) \in \mathbb{R}_+^{|J|+|\cup_{j\in J}\bar{P}_j|}$ be a solution to the LP relaxation of (SPP-LRSP), where $\bar{P}_j$ is the current set of pairings for facility $j \in J$.

Lemma 4.2 *If $n_R = 1$ or $n_R = |R|$, then $(\bar{t}, \bar{z})$ satisfies inequality (4.28).*

Proof If $n_R = 1$, inequality (4.28) is equal to a subtour elimination inequality, which is implied by the constraints of the set partitioning formulation (Letchford and González (2006)).

If $n_R = |R|$, then no two customers in R can be visited in the same route. Thus, if Lemma 4.1 is applied, the coefficient of pairing p that visits customers in set $K \subseteq R$ in (4.28) is at least $n_K = |K|$. If we sum the constraints (2.14) for all customers in set R, the validity of inequality (4.28) follows:

$$\sum_{j\in J}\sum_{p\in\bar{P}_j}\sum_{i\in R} a_{ip}\bar{z}_p = |R|,$$

$$\sum_{j\in J}\sum_{K\subseteq R}\sum_{p\in\bar{P}_j^K}\alpha_p\bar{z}_p \geq \sum_{j\in J}\sum_{K\subseteq R}\sum_{p\in\bar{P}_j^K}|K|\bar{z}_p = |R|.$$

□

If the minimum number of vehicle routes that are needed to serve customers in R is greater than 1 and less than $|R|$, then the rounded capacity inequalities may be violated by some fractional LP solutions of SPP-LRSP. An example of such a solution for a small instance follows:

Example Instance with four customers and two facilities:

- Let $I = \{1, 2, 3, 4\}$, $J = \{1, 2\}$.
- Demand is 20 for all customers. Vehicle capacity is 50, and facility capacity is 70.
- Let $(\bar{t}, \bar{z})$ be a solution to the LP relaxation of (SPP-LRSP).

- Nonzero variable values and the pairings represented by pairing variables are:

$$\begin{aligned}
\bar{t}_1 = \bar{t}_2 &= 1 \\
\bar{z}_1 &= 1 \quad : \quad \text{facility}_1 - 4 - \text{facility}_1 \\
\bar{z}_2 &= 0.5 \quad : \quad \text{facility}_2 - 3 - 1 - \text{facility}_2 \\
\bar{z}_3 &= 0.5 \quad : \quad \text{facility}_2 - 1 - 2 - \text{facility}_2 \\
\bar{z}_4 &= 0.5 \quad : \quad \text{facility}_2 - 3 - 2 - \text{facility}_2
\end{aligned}$$

- Rounded capacity inequality for $R = \{1, 2, 3\}$ is violated,

$$\sum_{i \in R} \sum_{k \in V \setminus R} x_{ik} = \sum_{j \in J} \sum_{p \in P_j} \sum_{i \in R} \sum_{k \in V \setminus R} l_{ikp} \bar{z}_p = 1.5 < 2.$$

Although it is possible to have an impact with these inequalities, we have decided to leave the implementation of these inequalities in our branch-cut-and-price algorithm as future work since we have reasons to believe that the probability of finding a violated rounded capacity inequality is low for the LRSP instances because of the multiple trips to the facility in a pairing. For a pairing p including multiple routes, $\sum_{i \in R} \sum_{k \in V \setminus R} l_{ikp} \geq \sum_{i \in R} \sum_{k \in V \setminus R} l_{ikq}$, where q is a pairing composed of single route. Consider the following solution based on the example presented before. The rounded capacity inequality for $R = \{1, 2, 3\}$ is not violated for this solution.

$$\begin{aligned}
\bar{t}_1 = \bar{t}_2 &= 1 \\
\bar{z}_1 &= 1 \quad : \quad \text{facility}_1 - 4 - \text{facility}_1 \\
\bar{z}_2 &= 0.5 \quad : \quad \text{facility}_2 - 3 - 1 - \text{facility}_2 - 2 - \text{facility}_2 \\
\bar{z}_3 &= 0.5 \quad : \quad \text{facility}_2 - 1 - 2 - \text{facility}_2 - 3 - \text{facility}_2
\end{aligned}$$

Next, we investigate some cutting planes used for the LRP, discuss their validity for the LRSP, and assess their impact with respect to the formulation (SPP-LRSP).

Chain Barring Inequalities. Laporte et al. (1986) introduced chain barring constraints to eliminate paths from one depot to another in a solution of the LRP. Let $\mathcal{P}_{\text{LRP}}$ be the LRP polytope based on a two-index formulation of the problem. Let $r = (j_1, i_1, i_2, i_3, .., i_k, j_2)$ be a path from node j_1 to j_2 where j_1 and j_2 are two depots and i_1, i_2, .., i_k are customer nodes. To eliminate path r, the following chain barring inequalities are defined:

$$x_{j_1 i_1} + 3x_{i_1 i_2} + x_{i_2 j_2} \leq 4, \qquad \text{if } k = 2, \forall x \in \mathcal{P}_{\text{LRP}}, \tag{4.29}$$

$$x_{j_1 i_1} + 2\sum_{t=1}^{k-1} x_{i_t i_{t+1}} + x_{i_k j_2} \leq 2k - 1, \quad \text{if } k \geq 3, \forall x \in \mathcal{P}_{\text{LRP}}. \tag{4.30}$$

We will prove that the LP relaxation of (SPP-LRSP) satisfies the following chain barring inequalities that are obtained by rewriting (4.29) and (4.30) in terms of the variables defining $\mathcal{P}_{\text{SPP}}$:

$$\sum_{p \in P_{j_1}} l_{j_1 i_1 p} z_p + 3\sum_{j \in J}\sum_{p \in P_j} l_{i_1 i_2 p} z_p + \sum_{p \in P_{j_2}} l_{i_2 j_2 p} z_p \leq 4, \qquad \text{if } k = 2, \tag{4.31}$$

$$\sum_{p \in P_{j_1}} l_{j_1 i_1 p} z_p + 2\sum_{j \in J}\sum_{p \in P_j}\sum_{t=1}^{k-1} l_{i_t i_{t+1} p} z_p + \sum_{p \in P_{j_2}} l_{i_k j_2 p} z_p \leq 2k - 1, \quad \text{if } k \geq 3, \tag{4.32}$$

$\forall (t, z) \in \mathcal{P}_{\text{SPP}}$. For ease of notation, let α_p be the coefficient corresponding to variable z_p for all $p \in \cup_{j \in J} P_j$. Let $(\bar{t}, \bar{z})$ be a solution to the LP relaxation of (SPP-LRSP) and in this solution $r = (j_1, i_1, i_2, i_3, .., i_k, j_2)$ be a path from depot j_1 to j_2 including k customers.

Lemma 4.3 *If pairing $p \in \cup_{j \in J} P_j$ visits exactly s customers in path r, then it can visit at most s arcs in path r. Depending on the number of customers in the path, the contribution (coefficient) of $\bar{z}_p$ to the left hand side of inequalities (4.31) and (4.32) is as follows:*

i. If $k = 2$, the contribution of p in (4.31) is $3(s-1) + 1 = 3s - 2$.

ii. If $k \geq 3$, the contribution of p in (4.32) is $2(s-1) + 1 = 2s - 1$.

Proof If p visits one customer in the path, then it can use at most one arc in r. If p only visits one of the customers in $\{i_2, .., i_{k-1}\}$, it cannot include any arc included in r. If p visits either i_1 or i_k, then it can include either the arc (j_1, i_1) or (i_k, j_2). The path cannot include both arcs since from the construction of p, p can only visit one of the facilities. Therefore, the contribution of variable $\bar{z}_p$ to both (4.31) and (4.32) can be at most one because of either (j_1, i_1) or (i_k, j_2).

If p visits s ($s > 1$) customers included in r, then it can use the largest number of arcs in r only if it visits these customers consecutively by using $s - 1$ arcs. And if p visits the first (i_1) or last customer (i_k) in the path, then it can also include either the arc (j_1, i_1) or (i_k, j_2). Therefore, p can use at most s arcs in r. The contribution of p to (4.31) and (4.32) follows from the coefficients in the inequalities and the fact that at most $s - 1$ of s arcs are between two customers. □

Proposition 4.2 *If a vector $(\bar{t}, \bar{z})$ satisfies (2.14) and (2.15) then it also satisfies the chain barring inequalities (4.31) and (4.32).*

Proof Let $r = (j_1, i_1, i_2, i_3, .., i_k, j_2)$ be a path from depot j_1 to j_2 including customers $i_1, i_2, .., i_k$ and assume that $\bar{z}$ violates either (4.31) or (4.32). Let $\bar{P}_j^s$ be the set of pairings visiting exactly s of the customers in r for facility j, for all $j \in J$ and $s = 1, .., k$.

We sum the constraints (2.14) for customers in set $R = \{i_1, i_2, .., i_k\}$:

$$\begin{aligned} \sum_{j\in J}\sum_{p\in\bar{P}_j}\sum_{i\in R} a_{ip}\bar{z}_p &= k, \\ \sum_{j\in J}\sum_{s=1}^{k}\sum_{p\in\bar{P}_j^s} s\bar{z}_p &= k. \end{aligned} \tag{4.33}$$

For $k = 2$, using lemma 4.3 part (i):

$$\begin{aligned} \sum_{j\in J}\sum_{p\in\bar{P}_j}\alpha_p z_p &\le \sum_{j\in J}\sum_{s=1}^{2}(3s-2)\sum_{p\in\bar{P}_j^s}\bar{z}_p = \sum_{j\in J}\sum_{p\in\bar{P}_j^1}\bar{z}_p + 4\sum_{j\in J}\sum_{p\in\bar{P}_j^2}\bar{z}_p \\ &\le 2(\sum_{j\in J}\sum_{p\in\bar{P}_j^1}\bar{z}_p + 2\sum_{j\in J}\sum_{p\in\bar{P}_j^2}\bar{z}_p) = 4. \end{aligned} \tag{4.34}$$

The equality in (4.34) follows from the equality (4.33) for $k = 2$. Therefore, (4.31) cannot be violated by any solution of the LP relaxation of (SPP-LRSP).

For $k \geq 3$, using lemma 4.3 part (ii), we have:

$$\begin{aligned}\sum_{j\in J}\sum_{p\in\bar{P}_j}\alpha_p z_p \;&\leq\; \sum_{s=1}^{k}(2s-1)\sum_{j\in J}\sum_{p\in\bar{P}_j^s}\bar{z}_p \\ &\leq\; \sum_{s=1}^{k}(s\frac{(2k-1)}{k})\sum_{j\in J}\sum_{p\in\bar{P}_j^s}\bar{z}_p \;\;=\;\; 2k-1. \qquad (4.35)\end{aligned}$$

The second inequality in (4.35) follows from the fact that $(2s-1) \leq s\frac{(2k-1)}{k}$ for all $s = 1..k$. To obtain the equality, multiply (4.33) by $(2k-1)/k$. The right hand side of (4.33) after the multiplication with $(2k-1)/k$ is equal to $2k-1$. Therefore, (4.32) cannot be violated by any solution of the LP relaxation of (SPP-LRSP). □

4.4 Disjunctive Valid Inequalities

In this section, we discuss the generation of disjunctive valid inequalities in the context of a branch-cut-and-price algorithm. As discussed in Section 1.2.5, many general-purpose valid inequalities (such as Gomory cuts and mixed integer rounding cuts) for mixed integer programming models can be derived based on disjunctions and can be considered to be disjunctive valid inequalities. Although generation of these general-purpose valid inequalities is very common in branch-and-cut algorithms, we haven't yet seen a study in the literature in which disjunctive valid inequalities are utilized in a branch-cut-and-price algorithm, with the exception of a recent presentation describing the application of disjunctive valid inequalities in a branch-cut-and-price algorithm developed for the capacitated VRP (Ropke (2008)). Here, we first describe how we can incorporate disjunctive valid inequalities into a branch-cut-and-price algorithm in general and then describe some procedures for generating disjunctive valid inequalities in the context of our branch-cut-and-price algorithm for the LRSP.

The theory behind and the main steps of the procedure to generate disjunctive valid inequalities are explained in Section 1.2.5. We briefly summarize the procedure here. Let (P) be the MILP for which we want to generate disjunctive valid inequalities. We refer to (P) as the *base model* in the cut generation procedure. The vector $\bar{x}$ is such that it is not contained in S (possibly $\bar{x}$ is a solution to the LP relaxation of (P)). In order to generate an inequality valid for S but violated by $\bar{x}$, we determine a disjunction valid for S but violated by $\bar{x}$. Here, we consider a disjunction with only two terms: $B^1\bar{x} \geq b^1 \vee B^2\bar{x} \geq b^2$. For a given base problem (P), a vector $\bar{x}$ and a binary disjunction, we construct the cut generating LP (CGLP) that is defined by constraints (1.3) - (1.6) and objective (1.2) and solve it. If the optimal solution value for the CGLP is negative, the result is an inequality (α, β) that is valid for S and violated by $\bar{x}$.

In a branch-cut-and-price algorithm, at any node of the tree (say node i), in order to generate disjunctive inequalities valid for S^i, we need to determine what our base model should be. At node i, we have only a restricted version of the subproblem P^i defined by a subset of columns C_G. If we try to generate a valid inequality based on the restricted version of P^i, we will have problems. First, it is not clear how we can find a disjunction valid for S^i in the cut generation procedure. Second, even if we are able to define a disjunction, the inequality generated based on the restricted version of P^i is not necessarily valid for S^i. We need to modify the inequality as we generate new columns, therefore the inequality would not have a certain structure and would be a group 3 inequality which is difficult to include in a branch-cut-and-price algorithm.

Instead of using a restricted version of P^i as a base model in the disjunctive cut generation procedure, we may use an alternative model that has a well-defined relationship with P^i. The basic idea that we develop is to use the alternative model as a base model to generate disjunctive valid inequalities and rewrite the generated inequality using the defined relationship between the alternative model and P^i. We have already presented several such models in previous sections of the thesis. Recall that we have derived the relationship between (G-LRSP) and (SPP-LRSP) by applying the DWD methodology to (G-LRSP) in Section 2.4. In Section 4.2.2, we present some reformulations of (SPP-LRSP) that have a small number of variables for which solution of the LP

relaxation does not require column generation. Later in this section, we define a reformulation of a relaxation of (G-LRSP) for which there is a well-defined relationship with (SPP-LRSP).

Let P_R^i be a reformulation of P^i. To simplify the presentation of our disjunctive cut generation procedure, we let $Y^i \equiv \Psi(x^i)$ denote the mapping of x^i (in the variable space of P^i) to Y^i (in the variable space of P_R^i), which is a set of equivalent solutions in the variable space of P_R^i. We use set Y^i to indicate that the mapping might be one-to-many, e.g., because of symmetry in the formulation of P_R^i. For $y \in Y^i$, we may generate a disjunctive valid inequality (α^y, β^y), maximally violated by y in the variable space of P_R^i and generated using the CGLP based on P_R^i. If $|Y^i| = 1$, we may rewrite (α^y, β^y), the valid inequality obtained by using $y \in Y^i$, as an inequality (α^x, β^x) valid for P^i. However, if $|Y^i| > 0$, we try to come up with a valid inequality (α^y, β^y) that is attained by combining the disjunctive valid inequalities obtained by using each member of set Y^i individually. For example, in a reformulation in which the variables are indexed by vehicle, such as formulation (G-LRSP), the fact that the vehicles are homogeneous is one reason for a one-to-many mapping. Requiring identical coefficients for vehicles located at the same facility solves this problem. To clarify this idea, we describe some implementations below which $|Y^i| > 0$.

In Figure 4.3, using this notation, we outline the main steps of the disjunctive cut generation procedure (referred to as DISJCUTGEN) for node i in the branch-cut-and-price algorithm. DISJCUTGEN generates a set of valid inequalities for a given instance of the problem defined based on the current set of columns and a fractional solution to its LP relaxation.

Next, we describe four implementations of the disjunctive cut generation procedure applied in our branch-cut-and-price algorithm for the LRSP. Each implementation chooses a different base model and different disjunctions. The details of each implementation are as follows.

Implementation 1. In this implementation, we use the reformulation of (SPP-LRSP) defined in Section 4.2.2. As we noted before, we use (ASPP-LRSP) in the branch-cut-and-price algorithm and we can rewrite and add valid inequalities (2.18) and (2.25) to the reformulation. The LP

DISJCUTGEN
Input: A subproblem P^i with column set C_G and x^i_{LP}.
Output: An inequality (α^x, β^x) valid for P^i and violated by x^i_{LP} if any.

1. Let P^i_R be the reformulation of P^i. Set $Y^i \leftarrow \Psi(x^i_{LP})$, where Y^i is the set of solutions in the variable space of P^i_R obtained by mapping Ψ.
2. Generate a valid inequality (α^y, β^y) by combining disjunctive valid inequalities obtained for y (using the CGLP based on P^i_R, y and some disjunction violated by y) for all $y \in Y^i$.
3. If (α^y, β^y) can be rewritten in the variable space of P^i, obtain (α^x, β^x). Otherwise, STOP.
4. If (α^x, β^x) is violated by x^i_{LP}, return the inequality. STOP.

Figure 4.3: Outline of the Disjunctive Cut Generation Procedure for Branch-Cut-and-Price

relaxation of the reformulation used in this implementation (referred to as (M_1)) is as follows:

$$
\begin{aligned}
\textbf{(}M_1\textbf{)} \qquad \sum_{j\in J} v_{ji} &= 1 && \forall i \in I,\\
\sum_{i\in I} D_i v_{ji} - L^F_j t_j &\le 0 && \forall j \in J,\\
\sum_{j\in J} t_j &\ge R, &&\\
v_{ji} - t_j &\le 0 && \forall j \in J, \forall i \in I,\\
0 \le t_j &\le 1 && \forall j \in J,\\
0 \le v_{ji} &\le 1 && \forall j \in J, \forall i \in I.
\end{aligned}
$$

Let $(\bar{t}, \bar{z}) \in \mathbb{R}_+^{|J|+\sum_{j\in J}|\bar{P}_j|}$ be a fractional solution to the LP relaxation of P^i at node i, where $\bar{P}_j$ is the set of pairings generated so far for facility $j \in J$. We can transform vector $(\bar{t}, \bar{v}) \in \mathbb{R}_+^{|J|+|J|\times|I|}$ in the variable space of (M_1) using $\bar{v}_{ji} = \sum_{p\in\bar{P}_j} a_{ip}\bar{z}_p$ for all $i \in I$, $j \in J$.

If $(\bar{t}, \bar{v})$ is fractional, we consider the following disjunctions in order to determine a disjunction violated by $(\bar{t}, \bar{v})$.

1. Flow from a facility to a customer must be either one or zero:

$$v_{ji} \leq 0 \ \vee \ v_{ji} \geq 1 \quad \forall j \in J, i \in I. \tag{4.36}$$

2. Total number of customers assigned to a facility must be an integer:

$$\sum_{i \in I} v_{ji} \leq g_j \ \vee \ \sum_{i \in I} v_{ji} \geq g_j + 1 \quad \forall j \in J, \tag{4.37}$$

   for some integer g_j and for all $j \in J$.

After discovering a violated disjunction of the form (4.36) or (4.37), we write the CGLP based on (M_1) and investigate a disjunctive valid inequality that is maximally violated by $(\bar{t}, \bar{v})$ in the form

$$\sum_{j \in J} \alpha_j^t t_j + \sum_{j \in J} \sum_{i \in I} \alpha_{ji}^v v_{ji} \ \geq \ \beta \quad \forall (t, v) \in \mathcal{P}^r_{\text{SPP}}. \tag{4.38}$$

The objective function of the CGLP is to minimize $\sum_{j \in J} \alpha_j^t \bar{t}_j + \sum_{j \in J} \sum_{i \in I} \alpha_{ji}^v \bar{v}_{ji} - \beta$. If the optimal solution value is negative, the CGLP results in a valid inequality $(\alpha, \beta) \in \mathbb{R}^{|J|+|J|\times|I|+1}$. We can rewrite this inequality in the variable space of $\mathcal{P}_{\text{SPP}}$ as

$$\sum_{j \in J} \alpha_j^t t_j + \sum_{j \in J} \sum_{p \in P_j} \sum_{i \in I} \alpha_{ji}^v a_{ip} z_p \ \geq \ \beta \quad \forall (t, z) \in \mathcal{P}_{\text{SPP}}. \tag{4.39}$$

A valid inequality in the form of (4.39) is a group 1 inequality and thus can easily be added to the branch-cut-and-price algorithm by modifying the costs of the incoming arcs for customer i by adding $\alpha_{ji}^v \omega$ for all $i \in I$ in the pricing algorithm for facility $j \in J$. ω is the dual variable corresponding to (4.39).

Implementation 2. In the second implementation of DISJCUTGEN, we chose a graph-based formulation that is a reformulation of a relaxation of (G-LRSP). In (SPP-LRSP), each pairing corresponds to a possible vehicle schedule associated with any vehicle. Thus, the vehicles are

not differentiated with an index. In light of this information, to be able to relate a generated valid inequality to (SPP-LRSP), we reformulate a relaxation of (G-LRSP) by mapping its variables with vehicle index to another variable space without the vehicle index. Let $u_{kij} = \sum_{h \in H_j} x_{kih}$ for all $(k,i) \in A$, $j \in J$, and $w_j = \sum_{h \in H_j} v_h$ for all $j \in J$. Variable u_{kij} will take value 1 if any vehicle located at facility j travels arc (k,i). w_j is the number of vehicles used at facility j. Let $\mathcal{P}^r_{\text{GBP}}$ denote the polytope defined by the feasible solutions to this reformulation. We use the following LP relaxation formulation referred to as (M_2) in the disjunctive cut generation procedure.

$$\textbf{(}M_2\textbf{)} \qquad \sum_{j \in J} \sum_{k \in V} u_{kij} = 1 \quad \forall i \in I, \tag{4.40}$$

$$-\sum_{i \in I} D_i \sum_{k \in V} u_{kij} + C^F_j t_j \geq 0 \quad \forall j \in J, \tag{4.41}$$

$$\sum_{k \in V} u_{ikj} - \sum_{k \in V} u_{kij} = 0 \quad \forall i \in I, j \in J, \tag{4.42}$$

$$TLw_j - \sum_{k \in V} \sum_{i \in V} t_{ki} u_{kij} \geq 0 \quad \forall j \in J, \tag{4.43}$$

$$0 \leq t_j \leq 1 \quad \forall j \in J, \tag{4.44}$$

$$0 \leq u_{ikj} \leq 1 \quad \forall i,k \in V, j \in J, \tag{4.45}$$

$$w_j \geq 0 \quad \forall j \in J. \tag{4.46}$$

As with (G-LRSP), constraints (4.40) require that each customer must be served. Constraints (4.41) are facility capacity constraints. Constraints (4.42) equate the number of vehicles in and out of a customer node. In contrast, (G-LRSP) has constraints (2.2) which are similar to constraints (4.42) but disaggregated by vehicle. We consider an aggregated version of (2.2) in the reformulation. Similarly, constraints (4.43) are the reformulation of the aggregated version of the time limit constraints (2.7) in (G-LRSP). While constructing (M_2), we dropped some constraints of (G-LRSP) that include the flow variables (represented by vector y). Therefore, (M_2) is a reformulation of a relaxation of (G-LRSP).

Variable y in (G-LRSP) indicates the amount of vehicle load between pairs of locations in the

problem. The reason for leaving out the constraints including flow variables is that we want to avoid generating valid inequalities including nonzero coefficients for flow variables. Based on the DWD procedure, we know how to rewrite the flow variables in terms of variables of (SPP-LRSP) using equation (2.31). However, we do not know how to handle valid inequalities of this type in the pricing algorithm. By using (M_2), we aim to generate disjunctive valid inequalities not including original flow variables or any reformulation of them. (M_2) enables a search for disjunctive valid inequalities of the form

$$\sum_{j\in J}\alpha_j^t t_j + \sum_{j\in J}\alpha_j^w w_j + \sum_{j\in J}\sum_{k\in V}\sum_{i\in V}\alpha_{kij}^u u_{kij} \;\geq\; \beta \quad \forall (t,w,u)\in \mathcal{P}^r_{\text{GBP}}. \tag{4.47}$$

Let $(\bar{t},\bar{z})\in\mathbb{R}_+^{|J|+\sum_{j\in J}|\bar{P}_j|}$ be a fractional solution for the LP relaxation of P^i. Define $(\bar{t},\bar{w},\bar{u})\in\mathbb{R}_+^{|J|+|J|+|J|\times|A|}$ using $\bar{w}_j=\sum_{p\in\bar{P}_j}\bar{z}_p$ and $\bar{u}_{kij}=\sum_{p\in\bar{P}_j}l_{kip}\bar{z}_p$ for all $(k,i)\in A$, $j\in J$.

If $(\bar{t},\bar{w},\bar{u})$ is fractional, then we consider the following disjunctions in order to determine a disjunction violated by $(\bar{t},\bar{w},\bar{u})$.

1. A facility is either open or closed.

$$t_j\leq 0 \;\vee\; t_j\geq 1 \quad \forall j\in J. \tag{4.48}$$

2. Total number of vehicles used at a facility is an integer.

$$w_j\leq g_j \;\vee\; w_j\geq g_j+1 \quad \forall j\in J, \tag{4.49}$$

   for some integer g_j.

3. Number of links into a set of customers is integer. Let S be a customer set such that $2\leq$

$|S| \leq 3$:

$$\begin{array}{rcl} \sum_{j\in J}(\sum_{k\in V\setminus S}\sum_{i\in S} u_{kij} + \sum_{k\in V\setminus S}\sum_{i\in S} u_{ikj}) & \leq & g \\ \vee & & \\ \sum_{j\in J}(\sum_{k\in V\setminus S}\sum_{i\in S} u_{kij} + \sum_{k\in V\setminus S}\sum_{i\in S} u_{ikj}) & \geq & g+1, \end{array} \tag{4.50}$$

for some integer g.

By solving the CGLP constructed using (M_2) and one of the disjunctions violated by $(\bar{t}, \bar{w}, \bar{u})$, we are searching for a valid inequality that is maximally violated by $(\bar{t}, \bar{w}, \bar{u})$ in the form (4.47). An optimal solution to the CGLP with negative objective value provides such an inequality $(\alpha, \beta) \in \mathbb{R}^{|J|+|J|\times|A|+1}$. We can rewrite the inequality in terms of variables of (SPP-LRSP).

$$\sum_{j\in J}\alpha_j^t t_j + + \sum_{j\in J}\alpha_j^w w_j + \sum_{j\in J}\sum_{p\in P_j}\sum_{(i,k)\in A}\alpha_{ikj}^u l_{ikp} z_p \geq \beta \ \ \forall (t,z) \in \mathcal{P}_{\text{SPP}}. \tag{4.51}$$

Note that we include w in (ASPP-LRSP) in order to allow implementation of one of our branching rules. A valid inequality in the form of (4.51) is a group 1 inequality. In order to use this class of inequalities in the branch-cut-and-price algorithm, in the pricing algorithm for facility j, we modify the cost of arc (i,k) with $\alpha_{ikj}^u \omega$ for all $(i,k) \in A$ and the total cost of a pairing with a fixed cost of $\alpha_j^w \omega$ for all $j \in J$, where ω is the dual variable corresponding to (4.51).

Implementation 3. In this implementation, the CGLP is formulated based on the LP relaxation of (G-LRSP). We refer to this LP relaxation as (M_3). Recall that (M_3) is as follows:

$$
\begin{aligned}
\textbf{(}M_3\textbf{)} \qquad \sum_{h\in H}\sum_{k\in N} x_{ikh} &= 1 \quad \forall i \in I, \\
\sum_{k\in N} x_{ikh} - \sum_{k\in N} x_{kih} &= 0 \quad \forall i \in N, h \in H, \\
\sum_{h\in H_j}\sum_{k\in I} y_{jkh} - L_j^F t_j &\leq 0 \quad \forall j \in J, \\
y_{ikh} - L^V x_{ikh} &\leq 0 \quad \forall (i,k) \in A, h \in H, \\
\sum_{k\in N} y_{ikh} - \sum_{k\in N} y_{kih} + D_i \sum_{k\in N} x_{ikh} &= 0 \quad \forall i \in I, h \in H, \\
\sum_{(i,k)\in A} T_{ik} x_{ikh} - L^T v_h &\leq 0 \quad \forall h \in H, \\
0 \leq t_j &\leq 1 \quad \forall j \in J, \\
0 \leq x_{ikh} &\leq 1 \quad \forall i,k \in V, h \in H, \\
0 \leq v_h &\leq 1 \quad \forall h \in H, \\
y_{ikh} &\geq 0 \quad \forall i,k \in V, h \in H.
\end{aligned}
$$

Let $(\bar{t}, \bar{z}) \in \mathbb{R}_+^{|J|+\sum_{j\in J}|\bar{P}_j|}$ be a fractional solution to the LP relaxation of P^i. Let $\hat{P}_j = \{p \in \bar{P}_j | \bar{z}_p > 0\}$ for all $j \in J$. We can generate a vector $(\bar{t}, \bar{v}, \bar{x}, \bar{y}) \in \mathbb{R}_+^{|J|+|J|\times|H|+|H|\times|A|+|H|\times|A|}$ associated with solution $(\bar{t}, \bar{z})$ using a mapping of (t, z) to (t, v, x, y). Since we have identical vehicles, we can assign pairings associated with nonzero variables from 1 to $|\hat{P}_j|$ for all $j \in J$ to vehicles in (M_3) arbitrarily, starting from the first vehicle. Let $(h \simeq p)$ represent such correspondence where p is the h^{th} smallest index in $\hat{P}_j$ and pairing p is assigned to vehicle h for all $h = 1, .., |\hat{P}_j|$ and $j \in J$. Note that a vector $(\bar{t}, \bar{z})$ may lead to multiple corresponding $(\bar{t}, \bar{v}, \bar{x}, \bar{y})$ vectors under the identical vehicles assumption. By defining the correspondence $(\simeq)$ between h and p, we restrict the mapping of vectors $(\bar{t}, \bar{z})$ and $(\bar{t}, \bar{v}, \bar{x}, \bar{y})$ to be one-to-one. To

construct $(\bar{t}, \bar{v}, \bar{x}, \bar{y})$, we use the following transformations based on $(\simeq)$:

$$\bar{v}_h = \begin{cases} \bar{z}_p & \text{If } (h \simeq p), \forall p \in \hat{P}_j, h \in H_j, j \in J, \\ 0 & \text{otherwise.} \end{cases}$$

$$\bar{x}_{ikh} = \begin{cases} l_{ikp}\bar{z}_p & \text{If } (h \simeq p), \forall p \in \hat{P}_j, h \in H_j, j \in J, \\ 0 & \text{otherwise.} \end{cases}$$

$$\bar{y}_{ikh} = \begin{cases} f_{ikp}\bar{z}_p & \text{If } (h \simeq p), \forall p \in \hat{P}_j, h \in H_j, j \in J, \\ 0 & \text{otherwise,} \end{cases}$$

where f_{ikp} is equal to the vehicle load on arc (i,k) for pairing p, for all $(i,k) \in A$, $p \in P_j$ and $j \in J$.

Using (M_3) and one of the disjunctions (4.48) - (4.50) that is violated by $(\bar{t}, \bar{v}, \bar{x}, \bar{y})$, we formulate the CGLP. Disjunction (4.50) needs to be rewritten in terms of x variables:

$$\sum_{j\in J}\Big(\sum_{k\in V\setminus S}\sum_{i\in S}\sum_{h\in H_j} x_{kih} + \sum_{k\in V\setminus S}\sum_{i\in S}\sum_{h\in H_j} x_{kih}\Big) \leq g$$

$$\vee$$

$$\sum_{j\in J}\Big(\sum_{k\in V\setminus S}\sum_{i\in S}\sum_{h\in H_j} x_{kih} + \sum_{k\in V\setminus S}\sum_{i\in S}\sum_{h\in H_j} x_{kih}\Big) \geq g+1,$$

for some set of customers S. The CGLP based on (M_3) enables a search for a disjunctive inequality valid $\forall (t,v,x,y) \in \mathcal{P}_{\text{GBP}}$ in the form of

$$\sum_{j\in J}\alpha^t_j t_j + \sum_{h\in H}\alpha^v_h v_h + \sum_{(k,i)\in A}\sum_{h\in H}\alpha^y_{kih} y_{kih} + \sum_{(k,i)\in A}\sum_{h\in H}\alpha^x_{kih} x_{kih} \geq \beta. \tag{4.52}$$

Since we cannot handle valid inequalities with nonzero coefficients for flow variables y, we add the following constraint to the CGLP to fix the coefficients associated with y to zero:

$$\alpha^y_{kih} = 0 \quad \forall (i,k) \in A, h \in H. \tag{4.53}$$

After a valid inequality in the form of (4.52) and satisfying (4.53) is generated for a given fractional solution, $(\bar{t}, \bar{v}, \bar{x}, \bar{y})$, we can rewrite more valid inequalities by circulating coefficients α^x_{ikh} and α^v_h over all $h \in H_j$ for all $j \in J$ based on the assumption of identical vehicles. In other words, we assume that we obtain multiple $(\bar{t}, \bar{v}, \bar{x}, \bar{y})$ vectors from one $(\bar{t}, \bar{z})$ vector (one-to-many correspondence between solution vectors). Then we can combine the coefficients α^x and α^v indexed by $h \in H_j$ for all $j \in J$ and rewrite the inequality (4.52) valid $\forall (t, v, x, y) \in \mathcal{P}_{\text{GBP}}$ in the form of

$$\sum_{j \in J} \alpha^t_j t_j + \sum_{j \in J} \alpha^{v_c}_j \sum_{h \in H_j} v_h + \sum_{j \in J} \sum_{(k,i) \in A} \sum_{h \in H_j} \alpha^{x_c}_{kij} x_{kih} \geq \beta^c \tag{4.54}$$

$$\begin{aligned} \text{such that} \quad \alpha^{v_c}_j &= \sum_{h \in H_j} \alpha^v_h, \\ \alpha^{x_c}_{ikj} &= \sum_{h \in H_j} \alpha^x_{ikh}, \\ \beta^c &= |I| \beta, \end{aligned}$$

where $|H_j| = |I|$ for all $j \in J$. Finally, we can rewrite this inequality in the (ASPP-LRSP) variable space:

$$\sum_{j \in J} \alpha^t_j t_j + \sum_{j \in J} \alpha^{v_c}_j w_j + \sum_{j \in J} \sum_{(i,k) \in A} \alpha^{x_c}_{ikj} \sum_{p \in P_j} l_{ikp} z_p \geq \beta^c \quad \forall (t, z) \in \mathcal{P}_{\text{SPP}}. \tag{4.55}$$

Inequalities in the form of (4.55) are in group 1 and can be easily used in the branch-cut-and-price algorithm by modifying the arc costs in the pricing algorithm.

In summary, we solve the CGLP to find a valid inequality in the form of (4.52) that is violated by $(\bar{t}, \bar{v}, \bar{x}, \bar{y})$. Then we modify the coefficients of (4.52) to get a valid inequality in the form of (4.54). Note that since the CGLP is solved for (4.52) and not for (4.54), the fractional solution $(\bar{t}, \bar{z})$ may not violate (4.55). In order to overcome this problem, we designed implementation 4.

Implementation 4. This implementation is designed to improve implementation 3. As in implementation 3, we use (M_3) to derive the disjunctive valid inequalities. The steps of this implementation are the same as those of implementation 3. However, we directly search for an inequality that is maximally violated by $(\bar{t}, \bar{v}, \bar{x}, \bar{y})$ and in the form of (4.54). Recall that implementation 3 searches for an inequality in the form of (4.52). In order to search for an inequality in the form of (4.54), we add constraints (4.53) to the CGLP. In addition, to be able to obtain the combined (aggregated for all vehicles located at the same facility) coefficients for x and v, we set the α coefficients for variables x_{ikh} and v_h to be equal for all $h \in H_j$ for $j \in J$.

$$\begin{aligned} \alpha^v_h &= \alpha^{v_c}_j, \quad \forall h \in H_j, \forall j \in J, \\ \alpha^x_{ikh} &= \alpha^{x_c}_{ikj}, \quad \forall h \in H_j, \forall j \in J. \end{aligned}$$

In Section 5.5.2, we provide computational results to evaluate the performance of each of these implementations and discuss some issues regarding.their effectiveness.

Chapter 5

Computational Experiments

In this chapter, we present computational experiments to empirically validate the procedures discussed in the previous three chapters. To begin, we first describe the details of the instances used in our experiments. We then present computational results related to Chapter 2 that compare the LP relaxation bounds for the two main formulations for the LRSP and assess the effect of the simple valid inequalities that are presented there. Next, we describe a set of experiments to evaluate a number of variants of the branch-and-price algorithms for the LRSP described in Chapter 3. Later, we discuss the effect of adding the procedures for dynamic cut generation described in Chapter 4. Finally, we present computational results that assess the cost benefits of solving the integrated LRSP as opposed to sequentially optimizing the location, routing and scheduling decisions.

In some of the experiments, we use performance profiles (introduced by Dolan and More (2002)) to aid visualization and evaluation of the performance of the algorithmic variants used in the experiments. Performance profile graphs consist of the cumulative distribution function for the ratio of a comparison criteria for an algorithm to the best corresponding value obtained in any of the algorithms in the pool. In our experiments, we generally use time and percentage gap as comparison criteria.

5.1 Description of the Instances

The only LRSP instances available in the literature are those in Lin et al. (2002) of which we were only able to obtain four small instances. We describe those instances and associated results in Section 5.4.6. Because they are very limited in size, we generated several sets of new instances for testing purposes. We describe these next.

Location and Demand Data. We generated three sets of instances with 20, 25 and 40 customers based on the MDVRP instances of Cordeau et al. (1997). All instances include five candidate facility locations. Each of our instances contains a subset of the customer locations and demand data from one of the Cordeau et al. (1997) instances. In particular, we selected instances $p01$ (50 customers), $p03$ (75 customers), and $p07$ (100 customers) from which to generate our three sets of instances.

Table 5.1 presents the sets of customers used to construct the LRSP instances in each group. The first column specifies the marker letter used to label the instances in our set (the details of how we label our instances is explained later in this section), the identity of the corresponding Cordeau et al. instance (denoted as *Ref*) and the indices of customers from the reference instance included in our instance. In addition to the customer indices, Table 5.2 lists the coordinates of the five facility locations in each instance group differentiated with each reference instance.

Travel Times. Rounded Euclidean distances between each pair of nodes were used in the algorithm as a proxy for travel times. Based on the assumption that the distances are in miles and vehicles travel 20 miles per hour, we can convert distances into travel times in order to justify the values of cost and time limit parameters with respect to real life data.

Capacity and Cost Parameters. Based on the demand information, we assumed that the capacity of all facilities was 250, which means that at least two facilities are required to be open in any feasible solution for each instance.

Table 5.1: Location & Demand Data for LRSP instances generated from Cordeau et al. (1997)

Marker Letter	Ref	Indices of the Cust. Coord. from Ref.		
		20-cust.	25-cust.	40-cust.
f	p01	1 - 20	1 - 25	1-40
l	p01	31 - 50	26 - 50	11-50
f	p03	1 - 20	1 - 25	1-40
m	p03	28 - 47	26 - 50	16-35, 41-60
l	p03	56 - 75	51 - 75	36-75
f	p07	1 - 20	1 - 25	1-40
s	p07	21 - 40	26 - 50	31-70
t	p07	31 - 50	51 - 75	1-20, 51-70

Table 5.2: Facility Locations for LRSP instances

Instance Group	Ref	Facility Coordinates (X,Y)
20- and 25-customer	p01	(20,20), (30,40), (15,40), (45,40) (55,50)
	p03	(40,40), (50,22),(25,45), (20,20), (55,55)
	p07	(15,35), (55,35),(35,20), (35,50), (25,45)
40-customer	p01	(20,20), (30,40), (15,40), (50, 30), (55,55)
	p03	(40,40), (50,22), (25,45), (20,20), (55,55)
	p07	(15,35), (55,35),(35,20), (35,50), (25,45)

For each instance, we considered two time limit values and two vehicle capacity values. Time limits of 7 and 8 hours (which can be converted to distance limits of 140 and 160 miles) were used. In all instances, 140 is more than twice the distance between each customer and its farthest candidate facility to ensure that any customer can be assigned to any facility, and the feasible solution set is not restricted because of an infeasible customer-facility pair. We used “t1” or “t2” to mark the instances based on the distance limit; “t1” was used for instances with distance limit 140 and “t2” was used for instances with distance limit 160 (the role of these markers in the labeling strategy used for our instances is explained later).

We chose two vehicle capacity values. To determine these values, we used the fact that at least two facilities are required to be open (due to the capacity) and used the following equation similar

to that used by Laporte et al. (1986):

$$B = (1 - \alpha)\text{max}_{i \in I}\{D_i\} + \alpha \frac{\sum_{i \in I} D_i}{2}, \tag{5.1}$$

with α equal to 0.1 and 0.2. Small α values were considered since a schedule employing multiple trips for a single vehicle is only realistic if the vehicle capacity is small compared to the demands of the customers. We used the 25-customer instances to determine vehicle capacities and used the same values in the 20- and 40-customer instances. For each α value, we took the average of the calculated values of B for the instances generated from the same Cordeau et al. (1997) instance (for example, all instances generated using $p01$), divided it by 10, rounded it up, and multiplied by 10. The purpose of averaging all instances that were generated from the same reference is to generalize the value of the parameter that is calculated, because the value of the equation (5.1) depends on the demand data and the demand data in instances that are generated using the same reference is from the same distribution.

We obtained vehicle capacity values 50 (for $\alpha = 0.1$) and 70 (for $\alpha = 0.2$) for the instances generated from $p01$ and $p07$, and 60 (for $\alpha = 0.1$) and 80 (for $\alpha = 0.2$) for the instances generated from $p03$. Vehicles with the smaller capacity values can cover at least one customer plus the customer that has the largest demand in the instance. We used "v1" or "v2" in the labels of the instances to differentiate the instances based on vehicle capacity.

Based on the cost values reported by Burlett (2002) and Kenworth Truck Company (2003), fixed costs for the facilities were generated using a uniform distribution on the interval [1500,2300], the fixed cost and the operating cost of a mid-size vehicle were estimated to be \$225 per day and \$1 per mile traveled, respectively. The same cost parameters were used in all instances.

Table 5.3 summarizes the capacity and cost parameters. In the table, as in Chapter 2, L^V and L^T represent the vehicle capacity and distance limit, respectively. C_j^F denotes the fixed cost of facility j for all $j \in J$, while C^V and C^O denote the vehicle fixed and operating costs, respectively.

Table 5.3: Summary of Capacity and Cost Parameters

Generated Using	L_1^V	L_2^V	L_1^T	L_2^T	Fac. cap.
p01,p07	50	70	140	160	250
p03	60	80	140	160	250

C_1^F	C_2^F	C_3^F	C_4^F	C_5^F	C^V	C^O
\$1615 /day	\$1845 /day	\$1781 /day	\$1975 /day	\$1753 /day	\$225 /day	\$ 1 /mile

Labels for the LRSP Instances. We used a labeling strategy to demonstrate the characteristics of each instance that was used in our computational experiments. We labeled our instances considering the reference Cordeau et al. (1997) instances, the set of customers included from the corresponding reference (the marker letters listed in Table 5.1 used to indicate this information), the number of customers, vehicle capacity (one of the markers v1 or v2) and distance limit values (one of t1 or t2). For example, we used label "p01-f25-v1t1" for the LRSP instance (in the 25-customer instance group) generated using customers 1 to 25 (based on letter "f") in $p01$. The "v1" and "t1" markers indicate that the smaller vehicle capacity and distance limit values (50 and 140, respectively, based on Table 5.3) were used in the instance. Another example label is "p03-m40-v1t2". This label denotes that the instance is one of the 40-customer instances and is generated from reference $p03$ using customers 16 to 35, 41 to 60. Vehicle capacity and distance limits are 60 and 160, respectively. Note that the only difference between instance "p03-m40-v1t1" and instance "p03-m40-v1t2" is the time limit value used.

5.2 Comparing the LP Relaxations of Alternative Formulations

The goal of the computational experiments in this section is to compare the LP relaxation bounds of (G-LRSP) and (SPP-LRSP). In Section 2.4, it is shown that the application of Dantzig-Wolfe Decomposition to the graph-based model (G-LRSP) results in the set partitioning-based model (SPP-LRSP), and using this result, in Section 2.4.2, we state that the LP relaxation bound of (SPP-LRSP) is at least as strong as the LP relaxation bound of (G-LRSP). In this section, we

validate this result and empirically demonstrate the strength of the LP relaxation of (SPP-LRSP) as compared to (G-LRSP).

We used the instances with 25 customers and 40 customers in this experiment. The LP relaxation of (G-LRSP) was solved using CPLEX 9.1. In (G-LRSP), the upper bound for the number of vehicles at each facility is the number of customers. However, this increases the number of variables and the LP solution times significantly. Therefore, we set a smaller upper bound value for the number of vehicles at each facility. For the 25-customer instances, we used the upper bound value 10 and for the 40-customer instances, we used the upper bound value 14. These maximum number of vehicle values are at least 40% larger than a simple estimate that can be calculated considering the facts that at least two facilities have to be open and vehicle capacity is large enough to cover at least two customers. These facts are not enough to decide a valid upper bound for the maximum number of vehicles at each facility. However, in the heuristic solutions to the LRSP instances obtained using I-Heur (see Section 3.4.1), the number of vehicles used at each open facility is much smaller than the upper bounds that we set. Therefore, we are confident that the number of vehicles used in the optimal solution does not exceed these bounds. The LP relaxation of (SPP-LRSP) (constraints (2.13) through (2.17)) was obtained by using the column generation algorithm E-COLGEN(0,0,50,350) described in Section 3.4.2. All instances in both experiment groups were run with a time limit of 8 CPU hours.

The results for the 25- and 40-customer instances are provided in Tables 5.4 and 5.5. The tables report the LP relaxation values and the solution times for (G-LRSP) and (SPP-LRSP) for each instance. Based on Tables 5.4 and 5.5, the LP relaxations obtained from (SPP-LRSP) are 5% to 10% stronger than the LP relaxations obtained from (G-LRSP). In addition, the computation times for (SPP-LRSP) are much better than the times for (G-LRSP) in all instances except one. For both the 25- and the 40-customer instances, solution times of (SPP-LRSP) are an order of magnitude better than solution times of (G-LRSP).

Table 5.4: LP Relaxation Bounds of (G-LRSP) and (SPP-LRSP): 25-Customer Instances

	(SPP-LRSP)		G-LRSP		(a-b)/b*100	
Data File	LP (a)	CPU (s)(c)	LP (b)	CPU (s)(d)	% Gap	c/d
p01-f25-v1t1	3959.89	8.28	3729.764	280.54	6.17	0.03
p01-f25-v1t2	3891.71	23.15	3664.293	272.44	6.21	0.08
p01-f25-v2t1	3836.89	20.88	3636.819	291.65	5.5	0.07
p01-f25-v2t2	3762.65	54.99	3575.846	299.38	5.22	0.18
p01-l25-v1t1	3749.46	3.52	3402.078	301.2	10.21	0.01
p01-l25-v1t2	3651.02	10.53	3329.01	316.47	9.67	0.03
p01-l25-v2t1	3593.77	11.08	3279.46	488.59	9.58	0.02
p01-l25-v2t2	3523.2	24.16	3214.12	426.4	9.62	0.06
p03-f25-v1t1	4633.06	4.75	4291.424	290.04	7.96	0.02
p03-f25-v1t2	4556.52	19.88	4217.155	347.01	8.05	0.06
p03-f25-v2t1	4465.42	8.95	4204.282	390.64	6.21	0.02
p03-f25-v2t2	4377.68	23.06	4136.160	513.38	5.84	0.04
p03-m25-v1t1	4481.74	11.43	4153.278	481.04	7.91	0.02
p03-m25-v1t2	4398.71	29.03	4085.718	285.73	7.66	0.1
p03-m25-v2t1	4334.34	27.31	4055.503	521.65	6.88	0.05
p03-m25-v2t2	4246.11	79.62	3994.734	466.8	6.29	0.17
p03-l25-v1t1	3951.49	4.36	3619.805	387.24	9.16	0.01
p03-l25-v1t2	3839.15	9.64	3545.993	343.01	8.27	0.03
p03-l25-v2t1	3826.99	10.32	3535.861	370.51	8.23	0.03
p03-l25-v2t2	3731.76	24.38	3466.029	348.45	7.67	0.07
p07-f25-v1t1	3418.42	7.86	3158.331	402.82	8.23	0.02
p07-f25-v1t2	3360.13	15.53	3092.431	285.36	8.66	0.05
p07-f25-v2t1	3317	22.38	3076.413	358.08	7.82	0.06
p07-f25-v2t2	3214.84	59.78	3014.934	260.57	6.63	0.23
p07-s25-v1t1	4136.93	1.98	3772.090	340.85	9.67	0.01
p07-s25-v1t2	4016.72	4.03	3688.345	471.77	8.9	0.01
p07-s25-v2t1	3966.49	4.58	3608.132	556.86	9.93	0.01
p07-s25-v2t2	3817.5	11.6	3534.823	561.57	8	0.02
p07-t25-v1t1	3754.74	5.97	3425.342	659.82	9.62	0.01
p07-t25-v1t2	3682.45	19.61	3359.481	621.15	9.61	0.03
p07-t25-v2t1	3631.42	12.31	3300.849	625.36	10.01	0.02
p07-t25-v2t2	3552.1	34.09	3240.682	374.81	9.61	0.09
Average		19.03		404.41	8.09	0.05

Table 5.5: LP Relaxation Bounds of (G-LRSP) and (SPP-LRSP): 40-Customer Instances

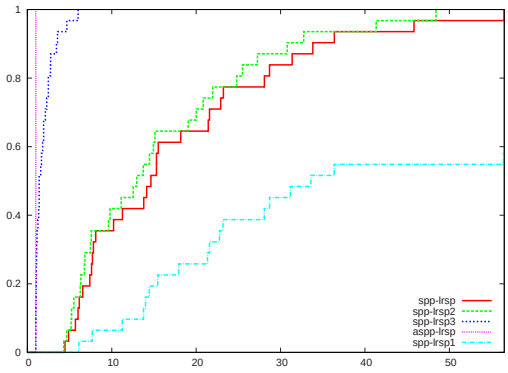
	SPP-LRSP		(G-LRSP)		(a-b)/b*100	
Data File	LP (a)	CPU (s) (c)	LP (b)	CPU (s) (d)	% Gap	c/d
p01-f40-v1t1	6019.5	139.06	5661.25	10701.91	6.33	0.01
p01-f40-v1t2	5940.94	613.25	5565.86	11683.6	6.74	0.05
p01-f40-v2t1	5851.26	475.53	5495.23	8037.53	6.48	0.06
p01-f40-v2t2	5750.61	1189.53	5406.40	13603.01	6.37	0.09
p01-l40-v1t1	5968.39	82.26	5575.07	11841.91	7.06	0.01
p01-l40-v1t2	5869.81	270.39	5479.82	11609.7	7.12	0.02
p01-l40-v2t1	5745.9	246.4	5423.90	13155.77	5.94	0.02
p01-l40-v2t2	5625.51	1297.07	5337.46	9286.94	5.4	0.14
p03-f40-v1t1	7255.42	273.61	6846.76	6190.5	5.97	0.04
p03-f40-v1t2	7137.05	1989.77	6748.45	6203.96	5.76	0.32
p03-f40-v2t1	7019.46	1121.65	6683.24	10388.86	5.03	0.11
p03-f40-v2t2	6945.08	5709.81	6595.75	14661.17	5.3	0.39
p03-m40-v1t1	6767.07	76.77	6380.18	10594.08	6.06	0.01
p03-m40-v1t2	6692.17	347.77	6281.08	10420.49	6.54	0.03
p03-m40-v2t1	6614.35	198.52	6222.73	6173.39	6.29	0.03
p03-m40-v2t2	6506.1	1177.49	6133.50	8741.29	6.07	0.13
p03-l40-v1t1	6208.81	154.05	5844.60	7394.12	6.23	0.02
p03-l40-v1t2	6127.31	350.96	5746.85	10487.14	6.62	0.03
p03-l40-v2t1	6034	324.94	5722.65	12068.38	5.44	0.03
p03-l40-v2t2	5942.4	2497.59	5633.57	10434.82	5.48	0.24
p07-f40-v1t1	5710.64	59.22	5308.25	5869.88	7.58	0.01
p07-f40-v1t2	5555.82	254.36	5203.41	5540.26	6.77	0.05
p07-f40-v2t1	5432.94	319.35	5123.49	8257.54	6.04	0.04
p07-f40-v2t2	5323.22	1303.25	5027.03	7140.7	5.89	0.18
p07-s40-v1t1	6113.67	32.63	5643.66	6115.72	8.33	0.01
p07-s40-v1t2	5949.65	75.07	5527.65	5337.76	7.63	0.01
p07-s40-v2t1	5801.4	88.14	5392.51	9352.83	7.58	0.01
p07-s40-v2t2	5661.45	380.95	5292.29	6786.14	6.98	0.06
p07-t40-v1t1	5482.41	205.45	5089.73	6323.65	7.72	0.03
p07-t40-v1t2	5369.25	639.62	4993.55	8390.05	7.52	0.08
p07-t40-v2t1	5223.99	2028.13	4916.37	7456.19	6.26	0.27
p07-t40-v2t2	5118.89	10607.65	4828.02	7506.42	6.02	1.41
Average		1079.07		8992.37	6.45	0.12

5.3 Assessing the Strength of Simple Valid Inequalities

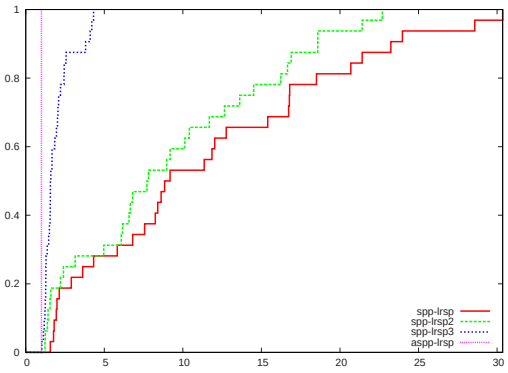
In this section, we assess the strength of the valid inequalities described in Section 2.3. Consider the base (SPP-LRSP) formulation containing only constraints (2.14) - (2.17). We construct (SPP-LRSP$_1$) by adding constraints (2.24), (SPP-LRSP$_2$) by adding constraints (2.25), (SPP-LRSP$_3$) by adding constraint (2.18), and (ASPP-LRSP) by adding both constraints (2.18) and (2.25) to (SPP-LRSP).

In this experiment, we solved the LP relaxations of formulations (SPP-LRSP), (SPP-LRSP$_1$), (SPP-LRSP$_2$), (SPP-LRSP$_3$) and (ASPP-LRSP) for 20-, 25- and 40-customer instances. We calculated the percentage LP relaxation gap associated with each formulations. We compared the LP relaxation gap of (SPP-LRSP) with the LP relaxation gaps of (SPP-LRSP$_1$), (SPP-LRSP$_2$) and (SPP-LRSP$_3$) to evaluate the strength of constraints (2.24), (2.25) and (2.18). Comparison between the results of (SPP-LRSP$_1$) and (SPP-LRSP$_2$) validates why we prefer constraints (2.25) to (2.24), and comparison between the results of (SPP-LRSP$_3$) and (ASPP-LRSP) evaluates the impact of adding constraint (2.18) alone and in combination with constraints (2.25). For the 20-customer instances, we enumerated all of the feasible pairings and solved the instances using a standard branch-and-bound algorithm without column generation. The reason for avoiding column generation and employing enumeration is that we could not incorporate constraints (2.24) into the column generation algorithm. For the 25- and 40-customer instances, we solved the LP relaxations of all formulations except (SPP-LRSP$_1$) using a column generation algorithm. Due to the large number of feasible pairings, we could not use enumeration for these instances, thus, we could not evaluate the effect of constraints (2.24) for the 25- and 40-customer instances.

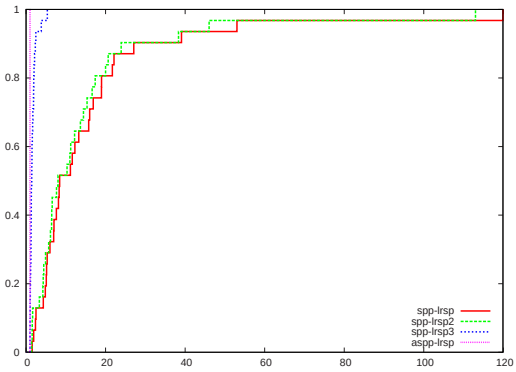
Figure 5.1 presents performance profiles for the LP relaxation gap for each of the formulations (SPP-LRSP), (SPP-LRSP$_1$), (SPP-LRSP$_2$), (SPP-LRSP$_3$) and (ASPP-LRSP). Performance profiles indicate that formulations (SPP-LRSP$_3$) and (ASPP-LRSP) perform significantly better than the others with respect to the LP relaxation gap. For all instances, (ASPP-LRSP) is the best



(a) 20-customer Instances



(b) 25-customer Instances



(c) 40-customer Instances

Figure 5.1: Performance profiles for LP relaxation gap of (SPP-LRSP), (SPP-LRSP$_1$), (SPP-LRSP$_2$), (SPP-LRSP$_3$) and (ASPP-LRSP)

formulation. The graphs show that in all instances, we can improve the LP relaxation gap with formulation (SPP-LRSP$_2$) compared to (SPP-LRSP), and Figure 5.1(a) shows that constraints (2.24) do not improve the performance of (SPP-LRSP).

In addition to the performance profiles, in Tables A.1, A.2 and A.3 in Appendix A, for each instance, we report the percentage LP relaxation gap associated with each of the formulations. For the 20-customer instances (in Tables A.1), we also report the total number of feasible pairings and the integer optimal objective value. For one instance, the set of feasible pairings could not be

generated, so we did not use it in the experiments. In Tables A.2 and A.3, we omited the column reporting optimal or best known objective value for each instance because we report these values later in this chapter.

Table A.1 indicates that adding inequalities (2.24) in (SPP-LRSP$_1$) made the problem difficult to solve and yielded at most only a small improvement in the LP relaxation bound for each solved instance. Based on Tables A.1, A.2 and A.3, adding inequalities (2.25) in (SPP-LRSP$_2$) improved the LP relaxation bound by a moderate amount. In all instances, adding inequality (2.18) in (SPP-LRSP$_3$) led to a dramatic reduction in the gap as compared to adding (2.25) alone. Together, the two inequalities (2.18) and (2.25) in (ASPP-LRSP) further reduced the gap for all but 5 instances. The average gaps for (SPP-LRSP$_3$) are 3.54, 4.47 and 2.96 in the 20-, 25- and 40-customer instances, respectively. These corresponding averages for (ASPP-LRSP) are 2.28, 2.68 and 2.04.

These results support the theoretical results in Section 2.3 and suggest that adding both inequalities (2.18) and (2.25) to the (SPP-LRSP) yields a stronger formulation which we refer to as (ASPP-LRSP) and used to obtain the results in the next section.

5.4 Computational Experiments With Branch and Price

The goal of the computational experiments in this section is to evaluate the effect of varying the individual components of the branch-and-price algorithm described in Chapter 3 and to explain the rationale for extensions of the basic branch-and-price algorithm used to improve the overall performance. In particular, we evaluate the different pricing algorithms used in the column generation process; the effect of modifying the column generation by using heuristic pricing algorithms; the strength of the primal heuristics to update the upper bound; and the effect of the method used to generate the initial set of columns. Finally, the overall performance of the branch-and-price algorithm is demonstrated for the 25- and the 40-customer instances, and some instances from literature.

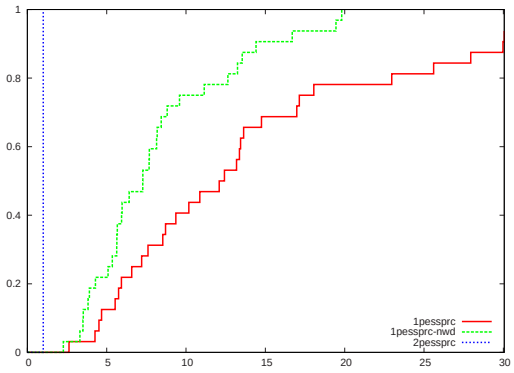
5.4.1 Performance of Exact Pricing Algorithms

In this section, we compare the performance of the exact pricing algorithms. The algorithms tested are the one-phase pricing algorithm (referred to as 1p-ESPPRC), the one-phase pricing algorithm with new domination rule (presented in Section 3.2.2), and the two-phase pricing algorithm (referred to as 2p-ESPPRC). We use notation 1p-ESPPR-NwD to denote pricing algorithm with new domination rule. In Section 3.2.2, we present the details of these pricing algorithms and we demonstrate empirically here that the new domination rule improves the performance of 1p-ESPPRC, and that 2p-ESPPRC performs better than 1p-ESPPR-NwD.

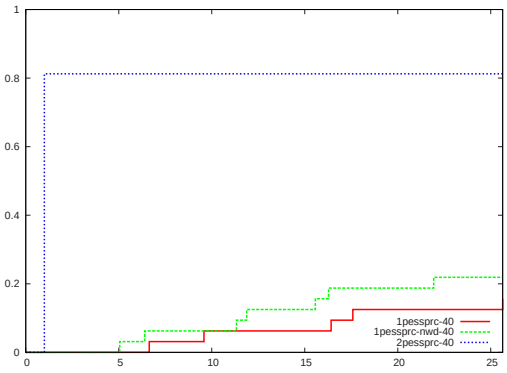
To compare the methods, we solved the LP relaxation of (ASPP-LRSP) at the root node using the column generation algorithm (COLGEN in Figure 1.4) with each one of the pricing algorithms: 1p-ESPPRC, 1p-ESPPRC-NwD and 2p-ESPPRC. In the experiments, at each iteration of COLGEN, for each facility, at most one column (the one with the most negative reduced cost) was added to the current restricted LP relaxation (RMP) of the problem. We ran the experiments with a time limit of 3 CPU hours and using the 25- and the 40-customer instances.

Figure 5.2 presents the performance profiles for the root node solution time using each of the pricing algorithms. The results clearly show that 2p-ESPPRC significantly outperforms both 1p-ESPPRC and 1p-ESPPRC-NwD. The effect of employing 2p-ESPPRC is even more significant for the 40-customer instances. Within the time limit of 3 CPU hours, COLGEN with 1p-ESPPRC and 1p-ESPPRC-NwD could solve the root node in fewer than 25% of the 40-customer instances, while 2p-ESPPRC could solve about 82% of the instances.

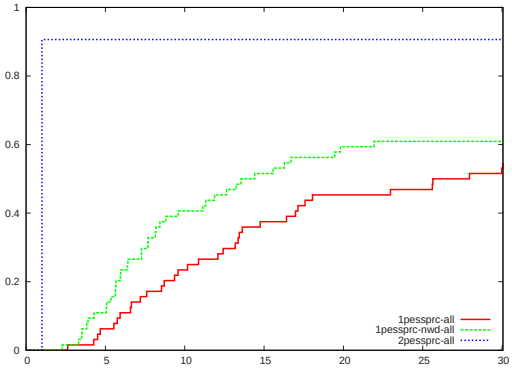
Tables A.4 and A.5 in Appendix A present details of the results from the experiments. For each 25- and 40-customer instance, and for each implementation of COLGEN, tables report the total CPU time (in seconds) to solve the LP relaxation of the problem (the root node solution time) and the average time for each COLGEN iteration (total COLGEN time / # of column generation iterations).



(a) 25-customer Instances



(b) 40-customer Instances



(c) 25- & 40-customer Instances

Figure 5.2: Performance profile for the root node solution time using 1p-ESPPRC, 1p-ESPPRC-NwD and 2p-ESPPRC

The results in Tables A.4 and A.5 show that for the instance set with 25 customers, the total COLGEN time was improved 36% on average with 1p-ESPPRC-NwD and 89% with 2p-ESPPRC, as compared to 1p-ESPPRC. The corresponding average values for the 40-customer instances solved within the time limit are 33% and 93%. In the 25-customer instances, the average time per iteration improved 43% with 1p-ESPPRC-NwD and 94% with 2p-ESPPRC. For 40-customer instances, the corresponding improvements were 36% and 70%, respectively.

In addition, we tested the effect of adding multiple columns at each iteration of COLGEN. At

each iteration of COLGEN and for each facility, we evaluated the addition of at most 1, 20, 40 and 100 of the columns with the most negative reduced costs. We use 2p-ESPPRC, 2p-ESPPRC$_{20}$, 2p-ESPPRC$_{40}$ and 2p-ESPPRC$_{100}$ to denote the exact pricing algorithms in which we add at most 1, 20, 40 and 100 of the columns per iteration, respectively. For these experiments, we used COLGEN with 2p-ESPPRC, 2p-ESPPRC$_{20}$, 2p-ESPPRC$_{40}$ and 2p-ESPPRC$_{100}$.

Figure 5.3 compares the root node solution times obtained with the variants of COLGEN employing 2p-ESPPRC, 2p-ESPPRC$_{20}$, 2p-ESPPRC$_{40}$ and 2p-ESPPRC$_{100}$. The performance profile indicates a performance improvement with the addition of multiple columns. The performance of each of 2p-ESPPRC$_{20}$, 2p-ESPPRC$_{40}$ and 2p-ESPPRC$_{100}$ are similar, however, though 2p-ESPPRC$_{40}$ is better by a slim margin.

Tables A.6 and A.7 in Appendix A present an instance-by-instance comparison of COLGEN with 2p-ESPPRC and 2p-ESPPRC$_{40}$. For each of these methods and for each instance, Tables A.6 and A.7 present the total CPU time (in seconds) to solve the root node, the number of COLGEN iterations, and the average time for each COLGEN iteration. Adding multiple columns at each iteration instead of one column reduced the number of COLGEN iterations in every instance. On average, for the 25-customer instances, the total time for 2p-ESPPRC$_{40}$ was 47% better than the time for 2p-ESPPRC. For the 40-customer instances, the time improvement was consistent, it was 50%, on average.

In summary, for the 25- and 40-customer instances, COLGEN with 2p-ESPPRC performed significantly better than the variants with 1p-ESPPRC and 1p-ESPPRC-NwD. In addition, the benefit of adding multiple columns instead of a single best column at each iteration of COLGEN was empirically demonstrated. In the rest of the experiments, we used 2p-ESPPRC$_{40}$ as exact pricing algorithm.

5.4.2 Effect of Using Heuristic Pricing Algorithms

In this section, we empirically demonstrate the performance changes for our branch-and-price algorithm with the addition of the heuristic pricing algorithms to the overall column generation

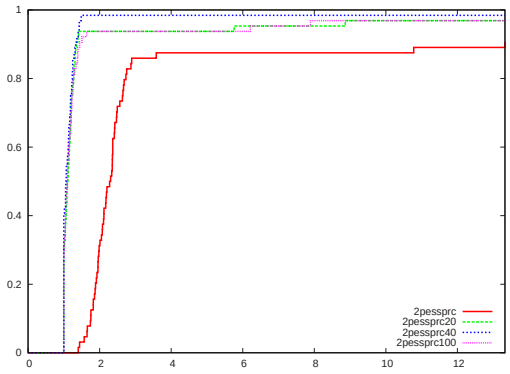


Figure 5.3: Performance profile for the root node solution time using 2p-ESPPRC and 2p-ESPPRC with multiple columns (20, 40 &100 columns per iteration)

scheme. In Section 3.2.3, we describe two heuristic versions of the pricing algorithm: ESPPRC with label limit (referred to as ESPPRC-LL) and ESPPRC for a subset of customers (referred to as ESPPRC-CS). In the preceding section, we observed that adding at most 40 columns at each iteration of the exact pricing algorithm resulted in the best performance. Therefore, we chose to add at most 40 columns at each iteration of the heuristic pricing algorithms. For ease of notation, we drop subindex 40 from the notation of the algorithms. The aim of this section is to evaluate the effect on the overall branch-and-price algorithm of employing 2p-ESPPRC-LL and 2p-ESPPRC-CS with 2p-ESPPRC in the column generation procedure.

Experiments Using the One-Stage Branch-and-Price Algorithm. In the computational experiments, we applied one-stage branch-and-price algorithms (1S-EBP) using the E-COLGEN procedure outlined in Figure 3.4 for a given set of input parameters. These parameters determine whether E-COLGEN calls 2p-ESPPRC-LL and 2p-ESPPRC-CS before 2p-ESPPRC and also includes the input parameters required for 2p-ESPPRC-LL and 2p-ESPPRC-CS themselves. Recall that E-COLGEN takes four parameters:

- u = the subset size in 2p-ESPPRC-CS,
- U = the maximum number of iterations for 2p-ESPPRC-CS,

- l = the initial label limit for 2p-ESPPRC-LL, and
- L = the label limit upper bound for 2p-ESPPRC-LL.

We refer to the algorithm with a given set of parameters as E-COLGEN(u, U, l, L). We compared four variants of 1S-EBP:

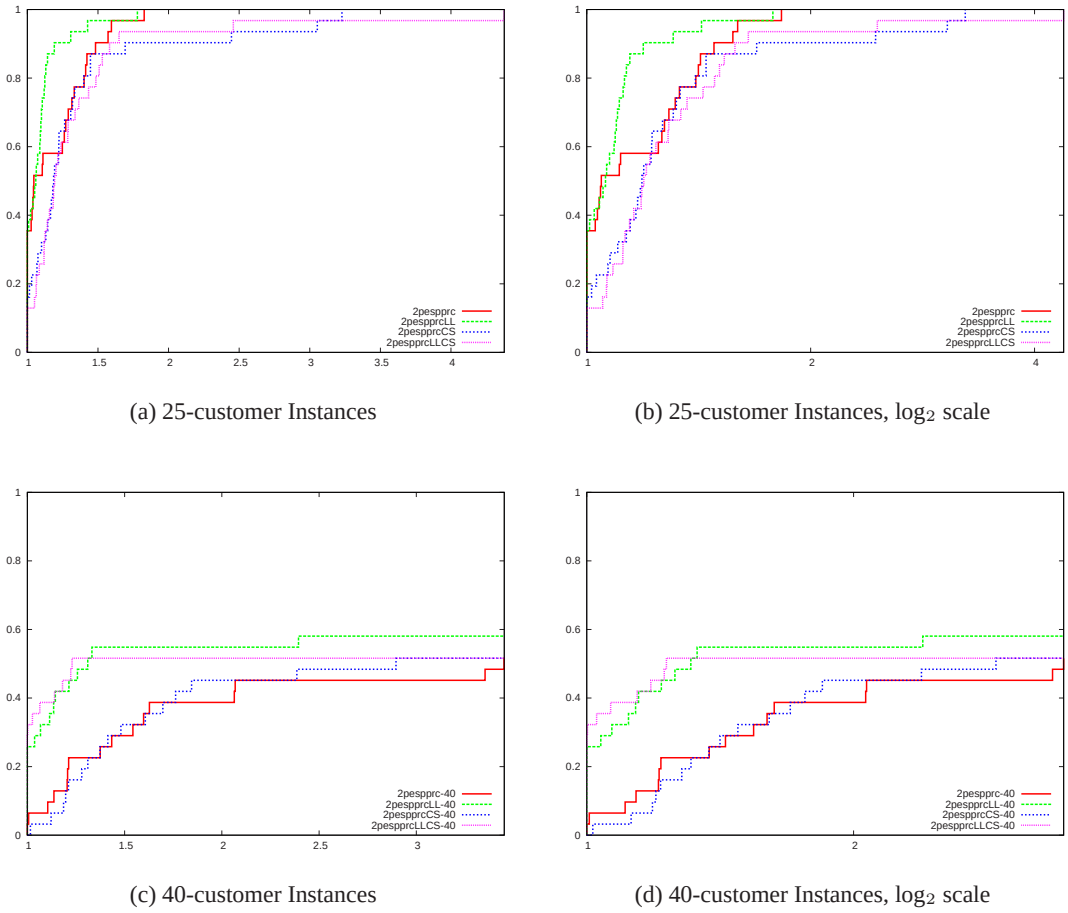
1. In the first experiment, we used 1S-EBP with E-COLGEN(0, 0, 0, 0), i.e., employing only exact pricing 2p-ESPPRC.
2. In the second experiment, we used 1S-EBP with E-COLGEN(0, 0, l, L), for some positive l and L, i.e., at each node, we only employed 2p-ESPPRC-LL before 2p-ESPPRC.
3. In the third experiment, we used 1S-EBP with E-COLGEN(u,U,0,0), for some positive u and U, i.e., we only employed 2p-ESPPRC-CS before 2p-ESPPRC.
4. Finally, we used both 2p-ESPPRC-CS and 2p-ESPPRC-LL before applying 2p-ESPPRC at each node of 1S-EBP.

In each experiment, we set a time limit of 8 CPU hours for each instance (with 25 and 40 customers). Based on preliminary computational experiments, we decided to set l= 50 and L = 350 in all experiments in this section whenever 2p-ESPPRC-LL was employed. The choice of $l = 50$ and L = 350 results in iterations of E-COLGEN in which 2p-ESPPRC-LL with label limits of 50, 100 and 200 is run. For small label limit values, 2p-ESPPRC-LL terminates quickly, but the generation of a column with the smallest reduced cost is less probable than 2p-ESPPRC-LL with larger label limit that can search a larger set of columns. In general, the number of columns with negative reduced cost is larger in early iterations of the column generation algorithm. Therefore, the number of labels at each node in the pricing algorithm tends to increase rapidly in the early iterations. Increasing the label limit gradually, we aim to pass the earlier iterations of E-COLGEN quickly, and explore a larger search space in later iterations to be able to get a column with smallest cost.

After some preliminary experiments, we decided to set U = 12 in all experiments, and $u = 12$ for the 25-customer instances and $u = 15$ for the 40-customer instances whenever 2p-ESPPRC-CS was employed. For large values of u, the heuristic becomes slow. However, if u is not large enough, the quality of the generated columns may not be sufficient (i.e., corresponding to pairings including less number of customers than they can actually include). Note that, the value of u differs based on the number of customers in the instance. We chose these values of u by looking at the solutions we found for the 25- and the 40-customer instances. We observed that a vehicle serves at most 11 customers in optimal solutions to the 25-customer instances, while a vehicle serves at most 14 customers in optimal or best solutions to the 40-customer instances. In addition, our experiments showed that U values greater than 20 reduced the overall performance of the algorithm.

Figure 5.4 presents the performance profiles for the total time spent in each variant of 1S-EBP: 1S-EBP that uses only the exact pricing algorithm (referred to as 2pespprc in the figure), 1S-EBP that also calls 2p-ESPPRC-LL (referred to as 2pespprcLL), 1S-EBP that calls 2p-ESPPRC-CS before 2p-ESPPRC (referred to as 2pespprcCS), and 1S-EBP that calls both 2p-ESPPRC-LL and 2p-ESPPRC-CS (referred to as 2pespprcLLCS).

Based on Figures 5.4(a) and 5.4(b), for the 25-customer instances, the variant that employs 2p-ESPPRC-LL outperforms all others. The variant that uses only 2p-ESPPRC seems to be better than the ones that call only 2p-ESPPRC-CS or both 2p-ESPPRC-CS and 2p-ESPPRC-LL. We conclude that the use of 2p-ESPPRC-CS does not improve the total solution time of the algorithm for the 25-customer instances. Tables A.8 and A.9 in Appendix A present some details of the experiments for the 25-customer instances. In Tables A.8 and A.9, for each instance and for each variant of 1S-EBP, we report the total time, # of evaluated nodes, the average COLGEN time (total COLGEN time / # of evaluated nodes). In addition, in both tables, we calculate the fraction of the total time in each variant to the total time of the one with only 2p-ESPPRC. For the 25-customer instances, Table A.8 shows that on average, applying 2p-ESPPRC-LL improves the total



(a) 25-customer Instances

(b) 25-customer Instances, $\log_2$ scale

(c) 40-customer Instances

(d) 40-customer Instances, $\log_2$ scale

Figure 5.4: Performance profile for the total time of 1S-EBP employing only 2p-ESPPRC or performing one or two of 2p-ESPPRC-LL and 2p-ESPPRC-CS before 2p-ESPPRC

execution time 5%, while employing 2p-ESPPRC-CS or both 2p-ESPPRC-CS and 2p-ESPPRC-LL increases solution time 18%, on average.

For the 40-customer instances, Figures 5.4(c) and 5.4(d) show that the variants of 1S-EBP that employ 2p-ESPPRC-LL outperform those without 2p-ESPPRC-LL. The variant that calls both 2p-ESPPRC-LL and 2p-ESPPRC-CS is better than the variant that calls only 2p-ESPPRC-LL in 55% of the instances. However, the variant that calls only 2p-ESPPRC-LL solves more instances within the time limit. Tables A.10 and A.11 in Appendix A present the total time; # of

evaluated nodes; the average COLGEN time; the integrality gap at the end of the algorithm; and the fraction of the total time in each version of 1S-EBP to the total time of the 1S-EBP without any heuristic pricing for each instance, and for each variant of 1S-EBP. On average, the addition of the 2p-ESPPRC-LL to the algorithm reduces the total time 13% and the addition of both 2p-ESPPRC-LL and 2p-ESPPRC-CS reduces the total time 16%. In instance p07-t40-v2t2, the 1S-EBP without any heuristic pricing routines could not even solve the root node within the time limit.

Overall, the computational results show that the addition of 2p-ESPPRC-LL to E-COLGEN in a branch-and-price algorithm improves the performance of the algorithm for both the 25- and 40-customer LRSP instances. Even though the addition of 2p-ESPPRC-CS does not cause any positive improvement for the 25-customer instances, we observed that the variant of 1S-EBP that includes both 2p-ESPPRC-LL and 2p-ESPPRC-CS performed well for the 40-customer instances. Therefore, we decided to investigate the performance of the 2p-ESPPRC-CS further in the larger LRSP instances, we implemented two sets of additional experiments using the instances with 40 customers: an analysis of the variants of the heuristic branch-and-price algorithm and the two-stage branch-and-price algorithm.

Experiments Using the Heuristic Branch-and-Price Algorithm. Based on the results of our previous experiments with the variants of 1S-EBP, we decided to investigate the effect of 2p-ESPPRC-CS in the heuristic branch-and-price algorithm framework described in Section 3.4.3. The overall algorithm is referred to as HBP and it is either used to provide an upper bound or to initialize the two-stage branch-and-price algorithm. In HBP, at each node of the branch-and-price tree, the heuristic column generation algorithm (H-COLGEN) presented in Figure 3.5 is used. We constructed two variants of HBP using two implementations of H-COLGEN:

1. In the first variant, apply only 2p-ESPPRC-LL as a pricing algorithm in H-COLGEN, i.e., use H-COLGEN(0, 0, l, L).

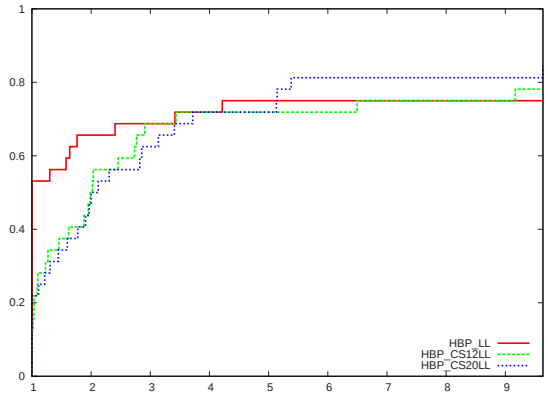


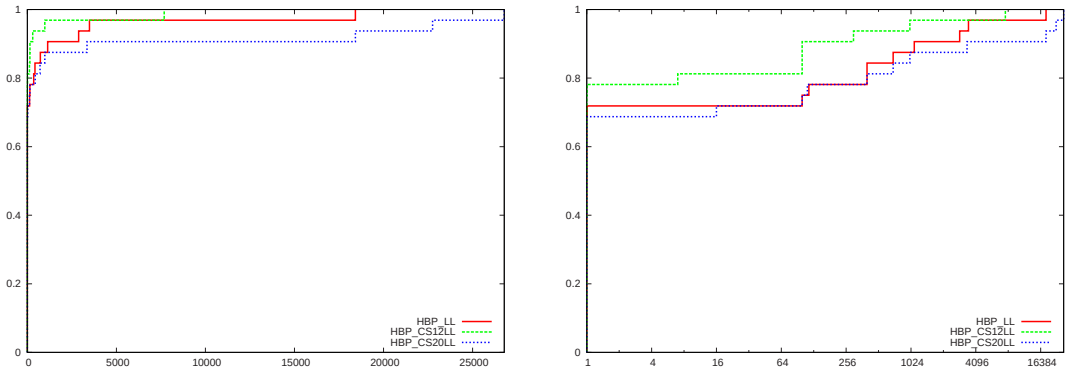
Figure 5.5: Performance profile for total time of HBP employing only 2p-ESPPRC-LL, and both 2p-ESPPRC-LL and 2p-ESPPRC-CS with $U = 12$ or 20

2. In the second variant, employ both 2p-ESPPRC-CS and 2p-ESPPRC-LL, i.e., use H-COLGEN(u, U, l, L).

In the experiments, we set $l = 5$ and L=15, and $u = 15$. We varied the second design by using two different values (12 and 20) for U. Recall that we set U=12 in 1S-EBP whenever we used 2p-ESPPRC-CS. By setting a higher value for U here, we hoped to get a better upper bound at the end of HBP at the expense of longer running times. HBP was run with a time limit of 2 CPU hours in each case.

Figure 5.5 compares the time spent in each variant of HBP. In the figure, HBP_LL refers to the first variant, HBP_CS12LL and HBP_CS20LL refer to the variants of HBP with 2p-ESPPRC-CS. The figure shows that HBP with only 2p-ESPPRC-LL is better than the others in 70% of the instances. However, the variants that employ 2p-ESPPRC-CS solve more of the instances within the time limit. Since the objective of applying HBP before an exact branch-and-price algorithm is to start with a good upper bound as well as a good set of columns, the quality of the upper bounds obtained in each variant of HBP is also important.

We evaluate the quality of the upper bounds in each HBP in Figure 5.6. We drew the performance profiles based on the percentage gap between the upper bound at the end of the algorithm



(a) 40-customer instances, HBP (%Gap)×100

(b) 40-customer Instances, HBP %Gap, $\log_2$ scale

Figure 5.6: Performance profile for the gap between z_{UB} & best known IP

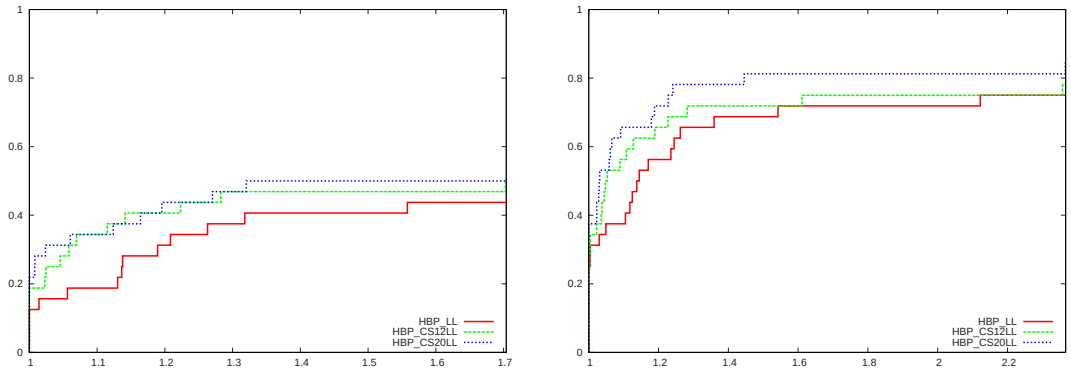
(z_{UB}) and the optimal or best known solution (IP). To generate the performance profiles, we replaced the 0% gaps with 0.0001% in order to avoid division by zero while calculating the ratio of performance measures. Figure 5.6(a) compares the 100×% gap, and Figure 5.6(b) compares the gap by using a $\log_2$ scale. Both figures show that the variant that uses 2p-ESPPRC-CS with U=12 performs slightly better than the others in terms of the quality of upper bound. Figures 5.5 and 5.6 indicate that we may prefer to use either the variant of HBP with only 2p-ESPPRC-LL or the variant with 2p-ESPPRC-CS in which U=12.

Table A.12 in Appendix A presents detailed results with three HBP variants: HBP with only 2p-ESPPRC-LL (columns 3 to 5); HBP with 2p-ESPPRC-CS and U set to 12 in addition to 2p-ESPPRC-LL (columns 6 to 8); and HBP that is the same as the second variant except that U is set to 20 (columns 9 to 11). In Table A.12, for each instance and for each HBP variant, we report the optimal or best known solution (Best IP), the upper bound that is obtained with each variant of HBP (z_{UB}), % gap between the z_{UB} and the best IP, and the total time. Based on the table, HBP with U = 12 results in the best average % gap and the execution time, however, there are some instances for which one variant of HBP results in a better gap or a lower execution time than the others.

Experiments Using the Two-Stage Branch-and-Price Algorithm. Next, we aim to investigate the effect of 2p-ESPPRC-CS in the two-stage branch-and-price algorithm (2S-EBP) framework described in Section 3.4.3. Our experiments here evaluate the variants of 2S-EBP similar to those of HBP evaluated in the previous set of experiments. The second stage of 2S-EBP is an 1S-EBP initialized using the upper bound and the columns from HBP. In particular, we used three different variants of HBP that were compared in the previous set of experiments as the first stages of a 2S-EBP algorithm. Thus, we obtained corresponding three variants of the 2S-EBP algorithm. We employed E-COLGEN(0, 0, 50, 350) in the second stage of each 2S-EBP algorithm. In our experiments, we used a time limit of 6 CPU hours in the second step. We did not employ 2p-ESPPRC-CS in E-COLGEN because of the results of our previous experiments. The aim of this set of experiments is to evaluate the effect of using 2p-ESPPRC-CS in addition to 2p-ESPPRC-LL in the first stage of 2S-EBP.

Figure 5.7 includes two sub-figures that show the performance of different 2S-EBP variants in terms of execution time. Figure 5.7(a) compares the total time for the second stage, while Figure 5.7(b) compares the total time for both stages. Figure 5.7 indicates that the variant that employs 2p-ESPPRC-CS in addition to 2p-ESPPRC-LL in the first stage performs better than the one that uses only 2p-ESPPRC-LL. The variant that uses 2p-ESPPRC-CS with U= 20 is slightly better than the one that employs 2p-ESPPRC-CS with U=12. For each instance, Table A.13 in Appendix A presents results for each variant of 2S-EBP.

Figure 5.8 evaluates the performance of the variants of the two-stage algorithms in terms of the final optimality gap. To generate the graph, we replaced the 0% optimality gaps with value 0.0001% and use a $\log_2$ scale as before. Based on the figure, the two-stage variants that use 2p-ESPPRC-CS perform better than the one that uses only 2p-ESPPRC-LL in the first stage. When we compare the optimality gap instance by instance (see Table A.13 in Appendix A), we observed that for most of the instances, the observed gaps are very similar for the variants of 2S-EBP, though the average gaps are slightly better in the two-stage algorithm with 2p-ESPPRC-CS. The best average is obtained with the design that sets U to 20.



(a) The 1S-EBP time in 2S-EBP

(b) Total 2S-EBP time

Figure 5.7: Performance profile for execution time of 2S-EBP

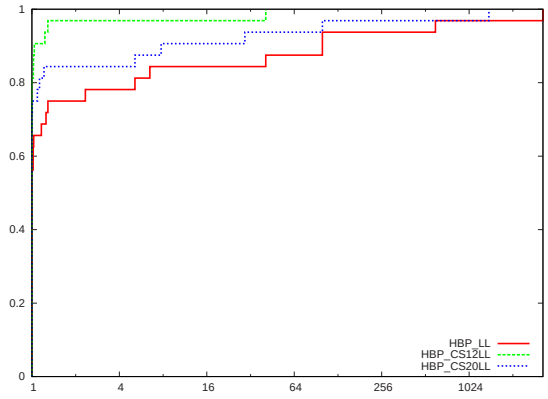


Figure 5.8: Performance profile for the optimality gap in 2S-EBP, $\log_2$

Overall, the test results showed that the addition of 2p-ESPPRC-LL to E-COLGEN improves the performance of the algorithm. The addition of 2p-ESPPRC-CS to the E-COLGEN, however, does not have a positive effect for the 25-customer instances. For the 40-customer instances, employing both 2p-ESPPRC-LL and 2p-ESPPRC-CS in E-COLGEN has good results. Furthermore, employing 2p-ESPPRC-CS in the first stage of the two-stage algorithm improves the performance.

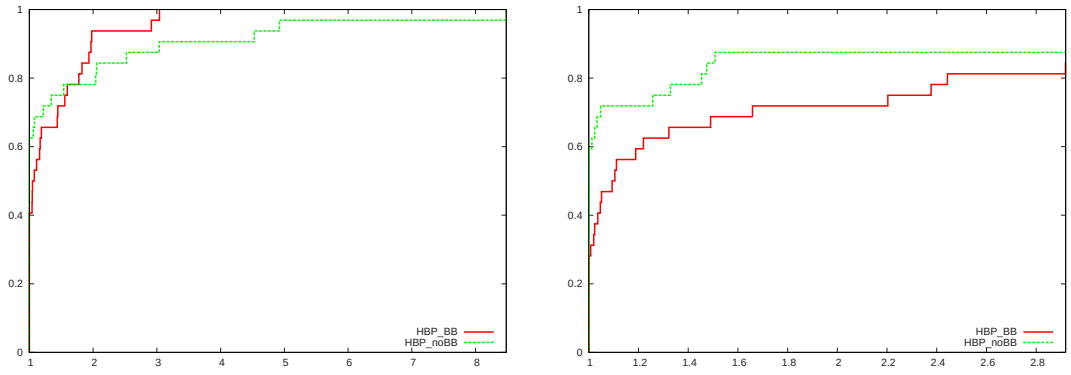
5.4.3 Performance of Heuristics

As described in Section 3.4, we implemented three approaches to finding primal feasible solutions: a simple construction-based heuristic (referred to as I-Heur and described in Figure 3.3); a standard branch-and-bound algorithm; and a heuristic branch-and-price algorithm, HBP (described in Section 3.4.3). In this section, we empirically demonstrate the quality of the upper bound obtained using each heuristic approach and compare the time spent in each approach. The branch-and-bound approach (referred to as BB) may be used at the root node of either HBP or 1S-EBP. In the BB approach, the IP model that includes columns which are present at the end of processing the root node of a branch-and-price tree is solved.

In the first experiment, we compared I-Heur, BB at the root node of HBP, HBP that calls BB at the root node and HBP that does not employ BB. We set a time limit of 1 CPU hour for BB and each of the two variants of HBP was run with a time limit of 2 CPU hours. In the HBP variant for the 25-customer instances, we used H-COLGEN(0, 0, 5, 15), and for the 40-customer instances, we used H-COLGEN(15, 20, 5, 15).

Figure 5.9 compares the execution times for the two variants of HBP. The figure indicates that for the 40-customer instances, the HBP without BB performs better than the HBP with BB. For the 25-customer instances, HBP without BB is slightly better for about 80% of the instances. Tables A.14 and A.15 in Appendix A compare the time spent and upper bounds obtained by each of the four heuristic approaches. The time used by I-Heur is significantly smaller than the other three approaches in all instances. The BB time which includes the root node solution time of HBP in addition to the solution time of an integer program, is smaller than the times spent in two variants of HBP. The average time in BB is about 85% better than the first HBP algorithm, while the second HBP time is 9% better than the first HBP variant.

In addition to the time, the strength of the upper bounds obtained by four heuristic approaches must be compared. We compared both the % gap between the LP relaxation and the z_{UB} and the % gap between the optimal (or best) solution and the z_{UB} obtained in each of the heuristics.



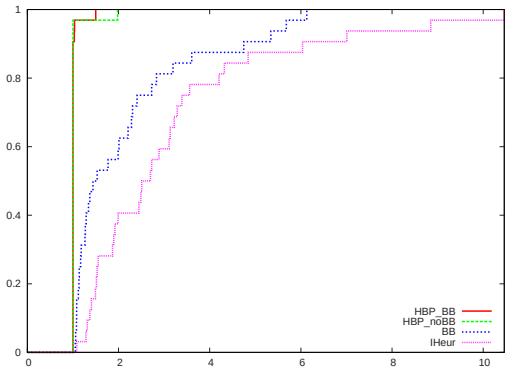
(a) 25-customer instances

(b) 40-customer instances

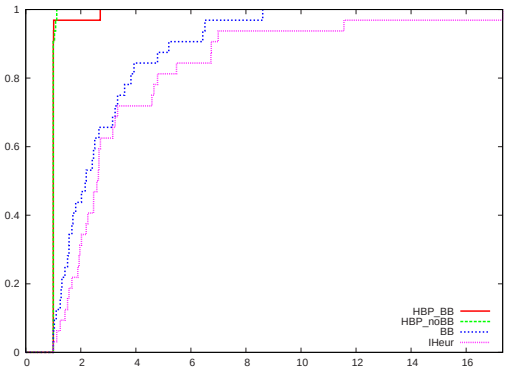
Figure 5.9: Performance profile for the execution time in HBP with and without BB

Figure 5.10 presents the performance profiles comparing these gaps; Figures 5.10(a) and 5.10(b) compare the % gap between the LP relaxation and the z_{UB} in the 25- and 40-customer instances, respectively, and Figures 5.10(c) and 5.10(d) compare the 100×%gap between the z_{UB} and the best IP. The two variants of HBP are significantly better than the other heuristics. Even though the performance of the HBP variants are very close to each other, the HBP without a BB seems to perform slightly better. In all instances, BB performs better than I-Heur.

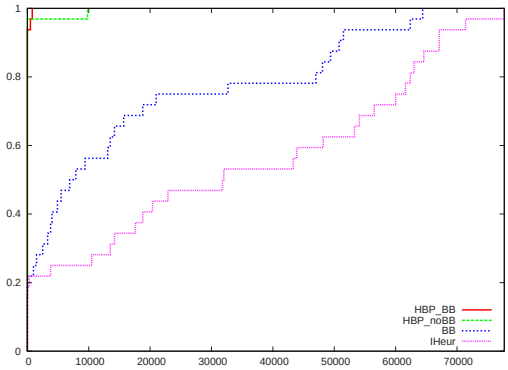
In addition to performance profiles, Tables A.16 and A.17 in Appendix A report the % gap between the optimal (or best) solution and the z_{UB} and % gap between the LP relaxation and the z_{UB}, for each variant and instance. As seen from Tables A.16 and A.17, I-Heur and BB provide upper bounds that are between 1.5% and 11.7% away from the LP relaxation bounds. In some instances, the BB bounds are slightly better than the I-Heur bounds. This difference is larger in the 25-customer instances; on average, BB provides bounds that are 5.5% away from the LP relaxation bound while the average gap is 7% in I-Heur. The two HBP variants provide stronger bounds than I-Heur and BB. The average gap between the upper bounds obtained in each of the HBP variant and the LP relaxation bounds is 2.9% in all instances, and the average values are similar in both 25- and 40-customer instances.



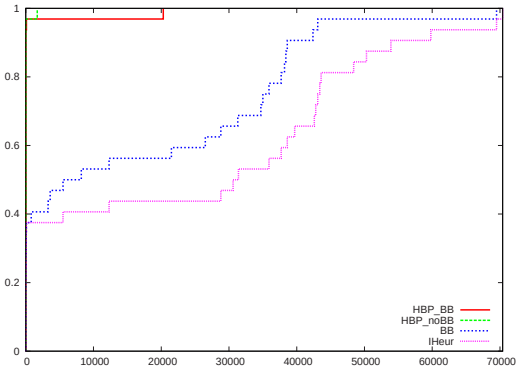
(a) Gap btw LP: 25-customer instances



(b) Gap btw LP: 40-customer instances



(c) 100×(%Gap btw IP): 25-customer instances



(d) 100×(%Gap btw IP): 40-customer instances

Figure 5.10: Performance profile for the gap between z_{UB}, and the LP bound and the optimal solution (IP)

Tables A.16 and A.17 also evaluate the strength of the primal heuristic bounds with respect to the best solution. I-Heur provides bounds that are about 4.6% from the optimal solution, on average. The effect of BB is more significant for the 25-customer instances compared to the 40-customer instances: the average gap for the 25-customer instances is 2.5%, while it is 3.6% for the 40-customer instances. Both HBP variants provide similar results and the upper bounds provided are very strong in almost all instances. The first HBP variant provides either optimal or best solution in 41 instances out of 64, while the second HBP variant provides either optimal or

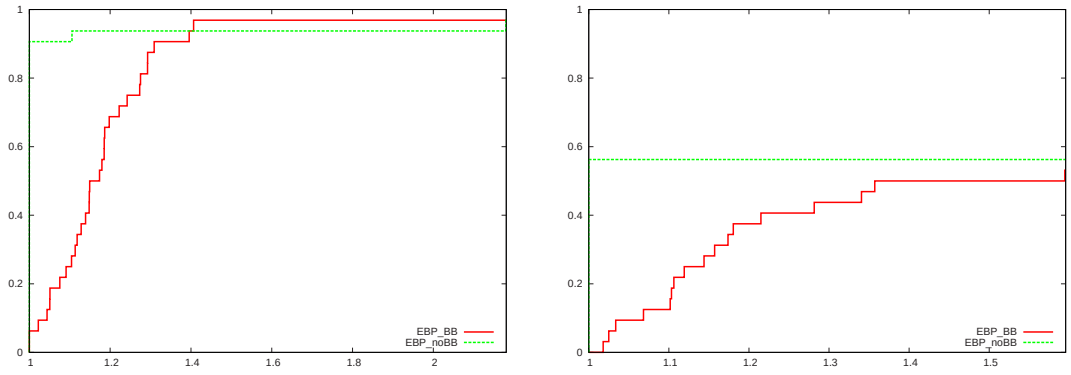
best solution in 43 instances. For both variants, the average gap between the optimal or the best solution is 0.5%.

These results suggest that all of the bounding procedures perform well and provide good upper bounds with respect to the LP relaxation or the best solution. The execution times for I-Heur are very small compared to the time limits set for the branch-and-price algorithm. Therefore, I-Heur is employed always. The execution times for the HBP variants are not very small, therefore, we can employ one of the HBP variants if the exact branch-and-price algorithm does not terminate very quickly, in particularly for larger instances. We evaluate the effect of HBP to the exact branch-and-price algorithm further in the next section. Since the two variants of HBP perform similar and the bounds obtained by BB are stronger than I-Heur we can employ BB at the root node of HBP. In this section, we evaluated BB at the root node of HBP. We may also employ BB at the root node of the exact branch-and-price algorithm, which we investigate next.

Evaluation of BB in One-Stage Branch-and-Price Algorithm. We designed another set of experiments to evaluate the effect of BB at the root node of the 1S-EBP algorithm: we compared 1S-EBP in which BB is employed at the root node with the 1S-EBP without BB. In both 1S-EBP variant, we used E-COLGEN(0,0,50,350) and set the time limit to 8 CPU hours.

Figure 5.11 presents the performance profiles based on the time spent in the variants of 1S-EBP for the 25- and 40-customer instances. The figure shows that 1S-EBP without BB outperforms 1S-EBP with BB with respect to the execution time. Figure 5.12 compares the variants of 1S-EBP with respect to the final optimality gap in the 40-customer instances. In more than 90% of the instances, the variant with BB provides better gaps than the one without BB.

Tables A.18 and A.19 present the results that compare the variants of 1S-EBP. Tables report the gap between the z_{UB} obtained using BB and the known optimal (or best) solution (column BBIP-Opt Gap), the number of evaluated branch-and-price nodes (column Nds) and the total time (column Time) for each variant of 1S-EBP, as well as the fraction of execution time in the variant with BB to the execution time of the variant without BB (column Time BB/WOBB). In addition,



(a) 25-customer instances

(b) 40-customer instances

Figure 5.11: Performance profile for the execution time of 1S-EBP with and without BB at the root node

Table A.19 reports the optimality gap at the end of each 1S-EBP algorithm (column Gap). Gap column is omitted from Table A.18 since in all 25-customer instances that are terminated normally, the optimality gaps are zero.

With the exception of two 25-customer instances, the total algorithm time increases in all 25- and 40-customer instances, if BB is employed at the root node. On average, the algorithm time increases about 13%. Even though the upper bound obtained using the BB procedure is about 1.8% away from the optimal (or best) solution on average, using BB at the root node of 1S-EBP does not improve the algorithm execution time.

Overall, in cases where the algorithm terminates because of the time limit, employing the primal heuristic BB at the root node of 1S-EBP reduces the optimality gaps; and in cases where the algorithm terminates with zero gap, the solution times are deteriorated with the BB procedure.

5.4.4 Comparing One- and Two-Stage Branch And Price

In this section, we compare the performance of the one-stage branch-and-price algorithm (1S-EBP) with the performance of the two-stage branch-and-price algorithm (2S-EBP). Recall that HBP is used to initialize 2S-EBP (see Section 3.4.3) and refers to the first stage of 2S-EBP where

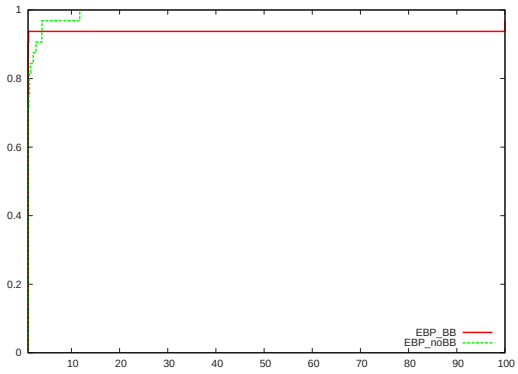
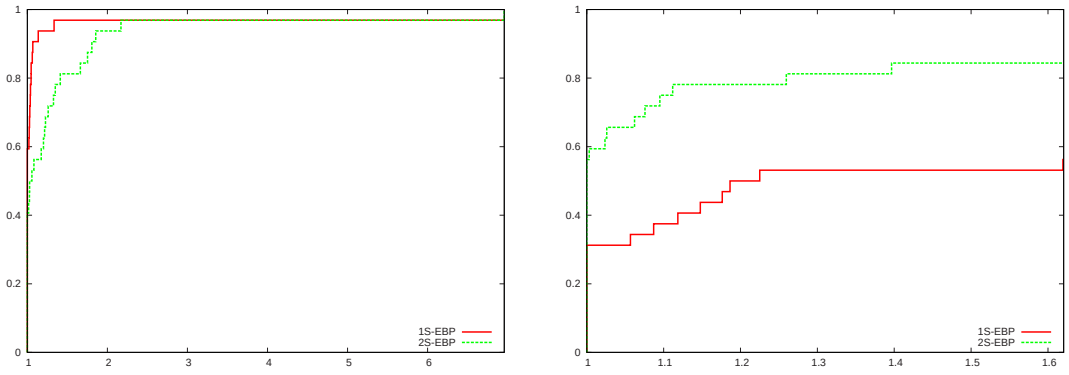


Figure 5.12: Performance profile for the optimality gap in 1S-EBP with and without BB for 40-customer instances

the second stage is an exact branch-and-price. Therefore, in this section, we evaluate the effect of employing HBP to the overall branch-and-price algorithm.

We implemented two experiments. In the first experiment, the one-stage branch-and-price algorithm employing E-COLGEN(0,0, 50, 350) was used. We set the time limit to 8 CPU hours. In the second experiment, we used the two-stage branch-and-price algorithm. In the first stage, we employed H-COLGEN(0,0,5,15) and set a time limit of 2 CPU hours. In the second stage, we initialized 1S-EBP with the columns and the upper bound from HBP and used 1S-EBP employing E-COLGEN(0, 0, 50, 350) with a time limit of 6 CPU hours. Thus, the total time limit for 2S-EBP was also 8 hours.

Figure 5.13 compares 1S-EBP and 2S-EBP based on the total solution time. In the 25-customer instances, 1S-EBP performs better than 2S-EBP, while in the 40-customer instances, 2S-EBP performs better than 1S-EBP. Figure 5.14 presents the performance profile comparing 1S-EBP and 2S-EBP with respect to optimality gap at the termination of the algorithm for the 40-customer instances. 2S-EBP is more successful in closing the gaps than 1S-EBP. Tables A.20 and A.21 present results of the experiments for the 25- and 40-customer instances, respectively, including the number of evaluated nodes, the total solution time in both 1S-EBP and 2S-EBP, and the execution time of 1S-EBP as a fraction of that of 2S-EBP. In the 25-customer instances, both



(a) 25-customer instances (b) 40-customer instances

Figure 5.13: Performance profile for the total solution time of 1S-EBP and 2S-EBP

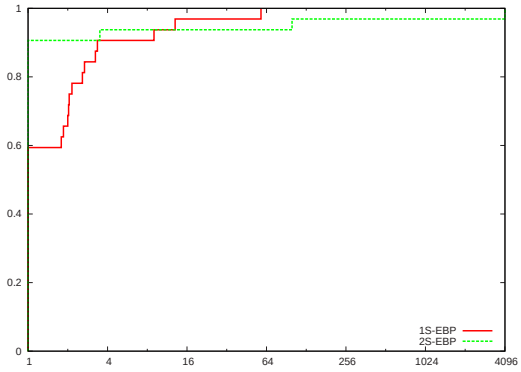


Figure 5.14: Performance profile for the optimality gap of 1S-EBP and 2S-EBP, $\log_2$ scale

algorithms terminated with optimal solutions in all cases. However, in some of the 40-customer instances, the algorithms terminated with a non-zero optimality gap. Thus, Table A.21 also reports the % gap at the end of the algorithm. Table A.20 indicates that on average, 1S-EBP is about 13% faster than 2S-EBP. However, based on Table A.21 1S-EBP is 7% slower than the 2S-EBP. If the optimality gaps for the 40-customer instances at the end of each algorithm 1S-EBP and 2S-EBP are compared instance by instance, 2S-EBP is more successful in closing the gaps. The strong upper bound obtained in HBP has a significant role in this. Therefore, for large instances, we prefer using 2S-EBP.

5.4.5 Performance on Generated Instances

In this section, we present solution and performance details of the exact branch-and-price algorithms for the 25- and 40-customer instances. Based on the results of our previous experiments, we propose below some default settings for our branch-and-price algorithm and present computational results for all 64 instances in the 25- and 40-customer instance sets.

Branch And Price for the 25-Customer Instances. For the 25-customer instances, we used 1S-EBP employing E-COLGEN(0,0, 50, 350) with a time limit of 6 CPU hours. Table 5.6 shows the results of our experiments. For each instance, we report the upper bound obtained from I-Heur (I UB), the LP relaxation value at the root node (LP), the optimal objective value (IP), the integrality gap at the end of the running time (Gap %), the number of evaluated nodes (Nds), and the total CPU time in minutes (CPU). The last three columns present some details about the optimal solution: the indices of open facilities (Loc), the number of vehicles used at each open facility (# Veh), and the total number of routes in each solution (# Rt).

Solution Time: Overall, the algorithm handled the 25-customer instances well; 29 of the 32 instances were solved to optimality in less than 25 minutes with the majority (24) solved in 10 minutes or less. For only one instance did the algorithm terminate due to the time limit before proving optimality. The strengths of the LP relaxation and I-Heur bounds have been discussed before in Sections 5.3 and 5.4.3, respectively.

Effect of Vehicle Capacity and Time Limit Parameters: As can be understood from the labels of the instances, the computational tests can also measure the sensitivity of the algorithm to vehicle capacity and time limit. As the labels of the instances indicate that there are 16 pairs of instances in which the pairs differ only in vehicle capacity (instances with label x-v1t1 and x-v2t1) and 16 pairs in which the pairs differ only in time limits (the instances with label x-v1t1 and x-v1t2). There are also eight instances in which both vehicle capacity and time limit increases from a reference instance (in an instance with label x-v2t2, both the vehicle capacity and the distance limit are higher than the instance x-v1t1). As expected, because the pricing problem gets more difficult, the

solution time tends to increase with the vehicle capacity and the vehicle time limit. In addition, the optimal cost does not increase when the vehicle capacity and/or the vehicle time limit increase(s). In 10 out of 16 pairs the computation time increases when the vehicle capacity increases, in 14 out of 16 pairs, the computation time increases when the distance limit increases. In all eight cases, the computation time increases when both time and vehicle limits increase. However, we note that there are also instances that favor the increase in time limit or capacity.

An increase in the vehicle capacity and time limit can reduce the cost through a change in the routing cost together with a decrease in the number of vehicles and/or a change in the set of selected facilities. On average, we calculated about 3.7% and 2.2% reduction in cost with vehicle capacity increase and time limit increase, respectively. The cost reduces about 5.8% on average if both vehicle and time limit increase.

Branch and Price for the 40-Customer Instances. For the 40-customer instance, we used 2S-EBP. HBP employing H-COLGEN(15, 20, 5, 15) was run with a time limit of 2 CPU hours. At the root node of HBP, BB was used. In the second stage of branch-and-price, we used E-COLGEN(0,0,50,350) and set a time limit of 6 CPU hours. Table 5.7 shows the results. For each instance, we report the upper bound obtained from I-Heur (I UB), the upper bound obtained at the end of HBP (HBP UB), the LP relaxation value at the root node (LP), the optimal objective value (IP), the integrality gap at the end of the running time (Gap %), the number of evaluated nodes (Nds), the CPU times spent in HBP and 1S-EBP, and the total CPU time in minutes.

Solution Time and Quality: As expected, the solution times in the 40-customer instances were larger than the 25-customer instances. Even though the algorithm terminated with a nonzero gap in some cases, the algorithm was able to prove optimality most of the time (21 of the 32 instance). For other instances, good solutions with small integrality gaps were obtained within the time limit. The gap was at most 0.35% for four of the instances and between 1.26% and 4.18% for the other seven instances. The strengths of the LP relaxation, I-Heur and HBP bounds have been discussed before in Sections 5.3 and 5.4.3. For 23 instances out of 32, the HBP upper bound value equaled

Table 5.6: Results for instances with 25 customers and 5 facilities

Data File	I UB	LP	IP[1]	Gap %	Nds	CPU (m)	Loc	# Veh	# Rt
p01-f25-v1t1	4795	4504.4	4699	0	25	1.3	0,4	2,2	10
p01-f25-v1t2	4795	4425.8	4511	0	297	208.1	0,4	1,2	10
p01-f25-v2t1	4722	4366.1	4425	0	99	9.9	0,4	1,2	7
p01-f25-v2t2	4722	4295.4	4425	0	119	15.9	0,4	1,2	7
p01-l25-v1t1	5193	4849.8	4899	0	24	0.6	0,1	2,2	8
p01-l25-v1t2	4968	4725.4	4815	0	139	2.9	0,4	2,2	8
p01-l25-v2t1	4809	4669.0	4759	0	157	5.7	0,4	2,2	6
p01-l25-v2t2	4762	4537.1	4602	0	39	6.9	0,1	1,2	7
p03-f25-v1t1	5281	4810.0	5062	0	236	3.0	0,2	3,2	10
p03-f25-v1t2	5139	4701.1	4837	0	77	2.5	0,2	2,2	10
p03-f25-v2t1	5048	4606.1	4758	0	393	4.6	0,4	2,2	8
p03-f25-v2t2	4831	4502.5	4742	0	52	5.3	0,4	2,2	8
p03-m25-v1t1	4995	4697.4	4785	0	509	5.2	0,4	2,2	10
p03-m25-v1t2	4869	4578.2	4785	0.1	3790	360.0	0,4	2,2	10
p03-m25-v2t1	4992	4485.8	4689	0	59	2.9	0,4	2,2	7
p03-m25-v2t2	4767	4396.1	4479	0	9	8.7	0,4	1,2	7
p03-l25-v1t1	5186	4795.3	4842	0	17	0.6	0,2	2,2	9
p03-l25-v1t2	4871	4670.2	4803	0	37	1.9	0,4	2,2	7
p03-l25-v2t1	4805	4620.9	4741	0	41	2.5	0,4	2,2	6
p03-l25-v2t2	4805	4507.6	4655	0	37	7.5	2,4	1,2	7
p07-f25-v1t1	4870	4642.6	4761	0	55	1.4	0,2	1,3	7
p07-f25-v1t2	4823	4544.5	4685	0	67	131.2	2,4	2,1	9
p07-f25-v2t1	4721	4485.6	4703	0	229	8.8	0,2	2,2	6
p07-f25-v2t2	4721	4388.3	4482	0	197	15.1	0,2	1,2	6
p07-s25-v1t1	5412	5000.1	5098	0	599	1.5	0,4	2,3	9
p07-s25-v1t2	5238	4852.2	4889	0	123	2.9	0,4	2,2	9
p07-s25-v2t1	5159	4739.4	4787	0	21	0.6	0,4	2,2	7
p07-s25-v2t2	4934	4609.8	4781	0	100	3.6	0,4	2,2	8
p07-t25-v1t1	5134	4836.5	4898	0	35	0.9	0,1	2,2	8
p07-t25-v1t2	5162	4703.1	4886	0	1345	25.0	0,2	2,2	8
p07-t25-v2t1	5025	4646.2	4767	0	777	18.8	0,2	2,2	6
p07-t25-v2t2	4769	4513.1	4592	0	91	12.6	0,1	1,2	6

[1]Optimal or best IP found by the algorithm within the time limit.

that of the optimal (or best integer) solution.

Effect of Vehicle Capacity and Time Limit Parameters: As with the 25-customer instances we consider the change in the performance of the algorithm with vehicle capacity and time limit values. In seven out of 16 pairs, the computation time increases when the vehicle capacity increases, in 12 out of 16 pairs, the computation time increases when the time limit increase. In six cases out of eight, the computation time increases when both time and vehicle limits increase. Because in a third of the 40-customer instances, the algorithm terminated due to the time limit, the solution times are not very sensitive to the changes in the capacity parameters. On average, we calculated about 3.8% and 1.6% reduction in cost with vehicle capacity increase and time limit increase, respectively. The cost is reduced about 5.3%, on average, if both vehicle and time limit increase. These results are in line with the corresponding averages for the 25-customer instances.

Discussion of the Solution Details. The last three columns of Table 5.6 and Table 5.9 present some details about the optimal solutions: the indices of open facilities (Loc), the number of vehicles used at each open facility (# Veh), and the total number of routes in each solution (# Rt) for the 25- and 40-customer instances, respectively.

In all 25-customer instances, the number of facilities selected to be used was two and the total number of vehicles used was between three and five, while the number of vehicles per facility was between one and three. In Table 5.8, we present some summary statistics about the solutions of all 32 instances, such as the number of vehicles used, the average number of routes per vehicle, the average number of customers per route and per vehicle, and the maximum number of customers per vehicle. Based on the table, we can see that each vehicle's schedule covers about two routes (2.1 routes per vehicle on average) and some vehicles cover more than three routes (average # of routes is between 1.5 to 3.3) including 11 customers at most. These statistics give us an intuition about the difficulty of the instances.

In all 40-customer instances, three to four facilities are selected to be open, and the total number of vehicles used changes between four to six. Similar to Table 5.8, Table 5.10 presents

Table 5.7: Results for instances with 40 customers and 5 facilities

	I	HBP			Gap		CPU(m)		
Data file	UB	UB	LP	IP[1]	%	Nds	HBP	EBP	Total
p01-f40-v1t1	7170	7131	6976.0	7131	0	807	53.2	150.0	203.2
p01-f40-v1t2	7170	7170	6826.2	6999	0.32	1065	120.0	360.0	480.0
p01-f40-v2t1	7064	6866	6700.2	6866	0	203	4.9	113.0	117.9
p01-f40-v2t2	6984	6821	6581.0	6821	1.26	35	35.4	360.7	396.1
p01-l40-v1t1	7449	7183	6899.7	7183	3.22	573	120.1	360.4	480.5
p01-l40-v1t2	7236	6934	6757.4	6934	0	43	5.6	48.0	53.6
p01-l40-v2t1	7166	6823	6616.7	6823	0	129	2.4	103.2	105.6
p01-l40-v2t2	7166	6633	6513.6	6612	0	25	8.4	187.6	196.0
p03-f40-v1t1	9253	8780	8697.4	8780	0	97	9.6	97.8	107.4
p03-f40-v1t2	9028	8753	8578.6	8753	0	41	28.0	154.6	182.6
p03-f40-v2t1	8943	8663	8439.0	8440	0	3	7.8	8.8	16.6
p03-f40-v2t2	8943	8438	8390.4	8438	0	9	11.5	138.2	149.7
p03-m40-v1t1	7827	7740	7050.9	7253	0	2524	120.0	360.1	480.1
p03-m40-v1t2	7544	7236	6907.7	7236	2.95	24	84.0	360.4	444.4
p03-m40-v2t1	7479	7132	6793.9	7132	2.79	27	120.0	360.1	480.1
p03-m40-v2t2	7430	6947	6655.5	6947	4.18	4	37.2	360.3	397.5
p03-l40-v1t1	7386	7167	6920.4	7167	0	183	1.5	55.6	57.1
p03-l40-v1t2	7212	6950	6771.6	6950	0.05	2121	120.0	360.0	480.0
p03-l40-v2t1	7121	6849	6662.9	6849	0	159	3.9	113.8	117.7
p03-l40-v2t2	7018	6846	6544.7	6639	0	53	39.0	290.9	329.9
p07-f40-v1t1	7424	7167	7053.1	7167	0	95	3.3	22.1	25.3
p07-f40-v1t2	7340	7001	6906.1	7001	0	433	11.0	75.5	86.5
p07-f40-v2t1	7174	6881	6745.5	6881	0	317	8.5	149.0	157.5
p07-f40-v2t2	7000	6845	6629.4	6845	2.34	43	20.3	360.1	380.4
p07-s40-v1t1	7860	7343	7256.3	7343	0	79	0.8	4.4	5.2
p07-s40-v1t2	7656	7124	7091.1	7117	0	29	1.8	6.5	8.3
p07-s40-v2t1	7466	7005	6924.5	7000	0	277	3.5	128.6	132.2
p07-s40-v2t2	7241	6944	6762.8	6944	0	251	5.7	211.2	216.9
p07-t40-v1t1	7283	7013	6940.5	7012	0	206	10.2	147.8	158.1
p07-t40-v1t2	7058	6972	6806.6	6972	0.35	66	76.2	360.5	436.7
p07-t40-v2t1	6945	6857	6649.8	6665	0.04	19	68.2	391.1	459.2
p07-t40-v2t2	6945	6658	6535.4	6658	1.64	6	26.1	363.3	389.5

[1]Optimal or best IP found by the algorithm within the time limit.

Table 5.8: More Solution Details for 25-customer instances

Case	Changes btw [min - max]	Avg.
# of vehicles per facility	1 - 3	
Total # of vehicles	3 - 5	
Avg. # of routes per vehicle	1.5 - 3.3	2.1
Avg. # of customers per route	2.5 - 4.2	3.3
Avg. # of customers per vehicle	5 - 8.3	6.9
Max # of customers per vehicle	7 - 11	8.5

some summary statistics about the solutions of the 40-customer instances. In the best solutions reported, each facility employed up to three vehicles, each of which was serving at least two routes and may serve up to four routes. On average, a vehicle covered 8.3 customers and the maximum number of customers per vehicle was between seven and 14.

The solution information provided in Tables 5.6 and 5.9 highlights the value of solving the integrated problem instead of a series of problems. Although the same number of facilities is selected within an instance group (meaning a fixed set of customer locations and a set of candidate facility locations), the particular facilities selected may differ depending on the vehicle capacity and time limit values. Which facilities are selected impacts which customers are served by each facility and hence the sequence of the routes and the structure of the pairings. From the solution information, we can see that, within an instance group, increases in the vehicle capacity and the time limit reduce the total cost through changes in facility, decreases in the number of vehicles required, and changes in the underlying route sequences.

In all 25- and 40-customer instances, only the minimum number of required facilities (based on the facility capacity) were selected to be open. This is expected because on average, the total facility fixed costs are 77% of the total cost. The second major cost component is the total vehicle fixed costs. On average, 17% of the total cost is the total vehicle fixed cost. Thus, the operating costs is about 6% of the total cost. In the 25-customer instances, the average vehicle fixed cost is about 17% of the average facility cost, while in the 40-customer instances, this percentage is about 25. In all instances, on average, the total operating cost is 8% of the total facility cost.

Table 5.9: Solution details for instances with 40 customers and 5 facilities

Data file	Loc	# Veh	# Rt	Data file	Loc	# Veh	# Rt
p01-f40-v1t1	0,3,4	2,2,1	14	p03-l40-v1t1	0,2,4	2,2,2	13
p01-f40-v1t2	0,2,4	1,2,2	14	p03-l40-v1t2	0,2,4	2,1,2	13
p01-f40-v2t1	0,2,4	1,2,2	11	p03-l40-v2t1	0,2,4	1,2,2	10
p01-f40-v2t2	0,3,4	2,1,1	10	p03-l40-v2t2	0,2,4	1,1,2	10
p01-l40-v1t1	0,2,4	1,2,3	15	p07-f40-v1t1	0,2,3	1,2,2	13
p01-l40-v1t2	0,2,4	1,2,2	15	p07-f40-v1t2	0,2,4	1,2,2	13
p01-l40-v2t1	0,2,4	1,2,2	10	p07-f40-v2t1	0,2,4	1,2,2	10
p01-l40-v2t2	0,2,4	1,1,2	10	p07-f40-v2t2	1,2,4	1,1,2	10
p03-f40-v1t1	0,1,2,4	2,1,1,1	16	p07-s40-v1t1	0,1,4	2,2,2	14
p03-f40-v1t2	0,1,2,4	1,1,2,1	16	p07-s40-v1t2	0,1,4	1,2,2	13
p03-f40-v2t1	0,1,2,4	1,1,1,1	11	p07-s40-v2t1	0,1,4	1,2,2	10
p03-f40-v2t2	0,1,2,4	1,1,1,1	11	p07-s40-v2t2	0,2,4	1,2,2	10
p03-m40-v1t1	0,2,4	2,2,2	15	p07-t40-v1t1	0,1,4	1,2,2	13
p03-m40-v1t2	0,2,4	2,2,2	15	p07-t40-v1t2	0,2,4	1,2,2	13
p03-m40-v2t1	0,2,4	2,2,2	11	p07-t40-v2t1	0,1,4	1,1,2	10
p03-m40-v2t2	0,1,4	2,1,2	12	p07-t40-v2t2	0,2,4	1,1,2	9

Table 5.10: More Solution Details for 40-customer Instances

Case	Changes btw [min - max]	Avg.
# of vehicles per facility	1 - 3	
Total # of vehicles	4 - 6	
Avg. # of routes per vehicle	1.8 - 3.2	2.5
Avg. # of customers per route	2.5 - 4.4	3.4
Avg. # of customers per vehicle	6.7 - 10	8.3
Max # of customers per vehicle	7 - 14	10.2

Table 5.11: Comparison of Lin et al. (2002) results and branch-and-price results

Instance		Lin B & B		Lin Heuristic		1S-EBP	
# Cust	# Fac	Objective	CPU (s)	Objective	CPU(s)	Objective	CPU(s)
10	4	309,817	1155	309,817	0.44	309,817	0.66
10	4	309,808	982	309,808	0.49	309,808	0.10
12	4	312,036*	$>$ 10,000	312,036	0.82	312,036	0.05
27	4	-	-	625,752.5	6	625,750.2	57.83

5.4.6 Performance on Instances from the Literature

The only LRSP instances available in the literature are those of Lin et al. (2002) which presents six instances: a base instance that includes 27 customers and four facility locations in Hong Kong plus five instances generated from the original by considering three subsets of 10 customers and two subsets of 12 customers while keeping the same four facility locations. We used only the four instances available to us for testing.

Lin et al. (2002) solved these instances with a branch-and-bound algorithm and with their heuristic algorithm. Within a time limit of 10,000 seconds, their branch-and-bound algorithm was able to prove optimality only for the 10-customer instances and was not able to find a feasible solution for the original 27-customer problem. The heuristic quickly found solutions for all instances. Table 5.11 compares their reported results to results obtained with 1S-EBP. As shown in Table 5.11, our algorithm quickly provided optimal solutions for all four instances. (The * indicates the best solution found in 10,000 CPU seconds on a Pentium III machine.)

5.5 Computational Experiments for Cut Generation

The goal of the computational experiments in this section is to test the performance of the branch-and-price algorithm when augmented with some of the cutting plane generation procedures described in Chapter 4. We empirically show the effect of each implemented procedure. We implemented lifted cover inequalities, generalized assignment polytope (GAP) cuts and disjunctive

valid inequalities.

5.5.1 Lifted Cover and GAP Inequalities

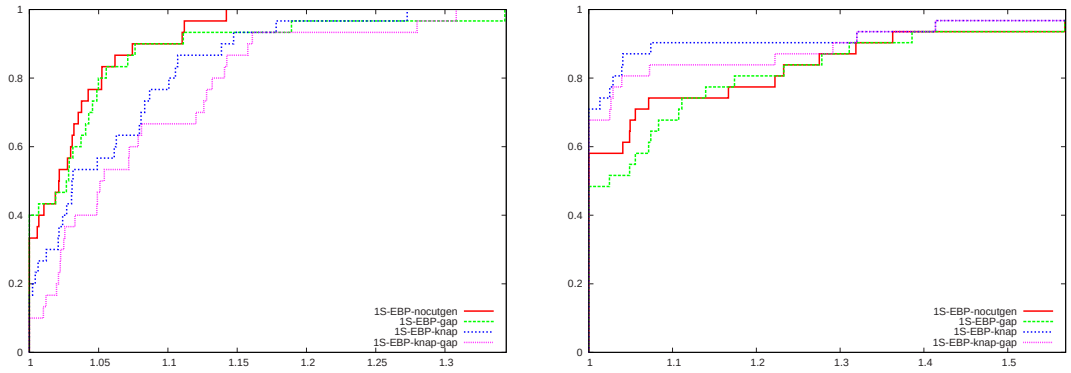
In this section, we evaluate the effect of lifted cover inequalities (inequalities of the form 4.16)) and GAP cuts (inequalities of the form (4.18)) which are valid for (SPP-LRSP) as described in Section 4.2.2.

For the experiments in this section, we solved the 25-customer instances with the one-stage branch-and-price algorithm (1S-EBP) using E-COLGEN(12, 12, 50, 350). In our implementations, to generate valid inequalities in 1S-EBP we ran cut generation procedure (referred to as CUTGEN) at each node after the column generation algorithm. For any node, we called CUTGEN if the depth of the node was less than 9. At node i of the branch-and-price algorithm, the steps of the bounding procedure were as follows:

- Call E-COLGEN.
- If x^i_{LP} is fractional and the depth $\leq$ 8, call CUTGEN procedure. Otherwise STOP.
- If a valid inequality is generated and added to P^i, call E-COLGEN procedure. Otherwise STOP.

We compared four variants of 1S-EBP:

1. 1S-EBP without any call to CUTGEN.
2. 1S-EBP employing the CUTGEN procedure that searches for violated lifted cover inequalities.
3. 1S-EBP employing the CUTGEN procedure that searches for violated GAP inequalities.
4. 1S-EBP employing the CUTGEN procedure that searches for both violated lifted cover and GAP inequalities.



(a) Total execution time

(b) # of evaluated nodes

Figure 5.15: Performance profile comparing 1S-EBP without any cuts, with CUTGEN$_K$, CUTGEN$_G$ and CUTGEN$_{K+G}$

We use notations CUTGEN$_K$, CUTGEN$_G$ and CUTGEN$_{K+G}$ to denote the cut generation procedures applied in the second, third and forth variants of 1S-EBP, respectively. We used a time limit of 6 CPU hours for all experiments.

Figure 5.15 compares the four variants of 1S-EBP based on the total solution time and the number of evaluated nodes in the branch-and-price tree in order to evaluate the effect of the CUTGEN procedures to the performance of the branch-and-price algorithm. Figure 5.15(a) indicates that 1S-EBP without any cuts and 1S-EBP employing CUTGEN$_G$ perform better than the other variants of 1S-EBP with respect to the execution time. CUTGEN$_G$ slightly improves the performance of 1S-EBP with respect to the execution time for some of the instances. However, we can conclude that the branch-and-price algorithm without any cut generation is better than the ones employing any of the cut generation procedures. Figure 5.15(b) indicates that the variants with lifted cover inequalities result in less number of node evaluations in the algorithm than the variant without any cut or the variant with CUTGEN$_G$.

Tables A.22 and A.23 in Appendix A present some details about the computational experiments comparing the four variants of the one-stage branch-and-price algorithm. Tables report the number of evaluated nodes (Nds) and the number of cuts generated (# cuts) in each variant of

1S-EBP. In addition, tables report the ratio of the time of a variant using a CUTGEN procedure to the time of the variant without CUTGEN (Time W/WO) to compare the performance of each variant with a CUTGEN against the variant without any cuts. Tables show that the CUTGEN procedure only effects the performance of the algorithm slightly. On average, the computational time of the variants with CUTGEN is almost the same as the variant without CUTGEN. On each instance, the time in the algorithms with CUTGEN is either slightly lower or slightly higher than the time for the base algorithm. In most of the instances, if the cut generation was successful, the number of evaluated nodes either stayed the same or decreased. Compared to 1S-EBP without any cut generation, in three out of 31 instances, the number of evaluated nodes increased in the variant with CUTGEN$_G$. These numbers were seven and six for the variants with CUTGEN$_K$ and CUTGEN$_{K+G}$, respectively.

As a summary, even though when we investigate the output in details, we can see that the LP relaxation bound improves slightly with the generated cuts at the relevant nodes and the total time spent in CUTGEN procedures is very small, the CUTGEN procedure does not contribute significantly to the performance of the overall algorithm.

We also tested the CUTGEN$_K$ and CUTGEN$_G$ in some of the 40-customer instances. We did not see promising results for larger instances either. In addition, we had some computational difficulties in most of the instances. Because of these new cuts, the size of the array that keeps the nonzero coefficients in constraints and cuts increases quickly and exceeds the predetermined limit of about 80000. We conclude that the improvement in the LP relaxation bound obtained with cuts is not worth the overall performance deterioration in the algorithm.

5.5.2 Disjunctive Valid Inequalities

In this section, we present some computational results for the four implementations of the disjunctive cut generation algorithm described in Section 4.4. We refer to the cut generation procedures for each of the four implementations as DISJCUTGEN$_{I1}$, DISJCUTGEN$_{I2}$, DISJCUTGEN$_{I3}$, and DISJCUTGEN$_{I4}$.

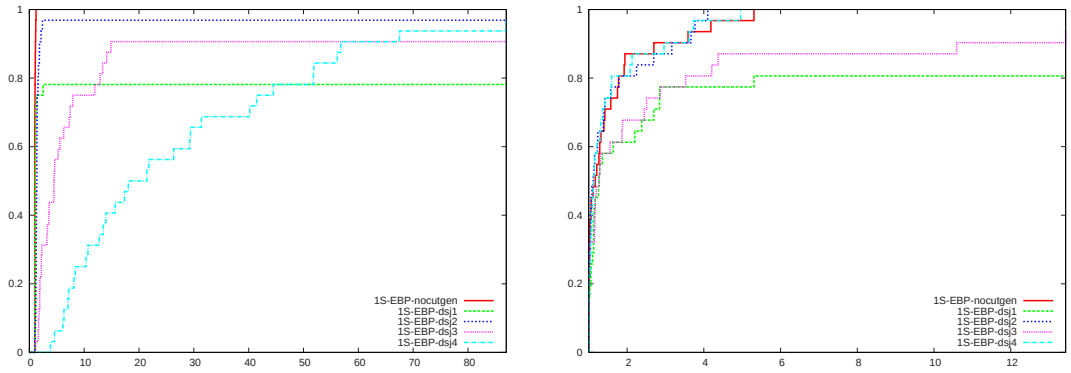
For the experiments in this section, we solved the 25-customer instances with the one-stage branch-and-price algorithm (1S-EBP) using E-COLGEN(12, 12, 50, 350). We used a time limit of 6 CPU hours. At each node, when the depth was 8 or less, we called one of the DISJCUTGEN procedure after the column generation algorithm. The steps of the bounding procedure followed at each node of the branch-and-price tree were the same as those listed in the previous section except that we called a DISJCUTGEN procedure. All disjunctive valid inequalities generated were added as local for the corresponding node.

We compared five variants of 1S-EBP:

1. 1S-EBP without any call to DISJCUTGEN.
2. 1S-EBP employing DISJCUTGEN$_{I1}$ in order to identify violated disjunctive valid inequalities based on implementation 1.
3. 1S-EBP employing DISJCUTGEN$_{I2}$ in order to identify violated disjunctive valid inequalities based on implementation 2.
4. 1S-EBP employing DISJCUTGEN$_{I3}$ in order to identify violated disjunctive valid inequalities based on implementation 3.
5. 1S-EBP employing DISJCUTGEN$_{I4}$ in order to identify violated disjunctive valid inequalities based on implementation 4.

In variants with DISJCUTGEN$_{I2}$, DISJCUTGEN$_{I3}$ and DISJCUTGEN$_{I4}$, we added at most one disjunctive cut at each node. In the branch-and-price algorithm employing DISJCUTGEN$_{I1}$, at each node, we may generate at most four disjunctive valid inequalities.

Figure 5.16 compares the performance of five variants of 1S-EBP based on the execution time and the number of evaluated nodes. Figure 5.16(a) indicates that we obtain the best performance (with respect to the execution time) with the one-stage branch-and-price without any cut generation calls. However, the performance of the variants employing DISJCUTGEN$_{I1}$ and DISJCUTGEN$_{I2}$ are relatively close to the variant without cut generation calls. DISJCUTGEN$_{I4}$



(a) Total execution time

(b) # of evaluated nodes

Figure 5.16: Performance profile comparing 1S-EBP without any cuts, with DISJCUTGEN$_{I1}$, DISJCUTGEN$_{I2}$, DISJCUTGEN$_{I3}$, DISJCUTGEN$_{I4}$

procedure reduces the performance of 1S-EBP significantly compared to other cut generation procedures in terms of computation time. Figure 5.16(b) indicates that the variants with DISJCUTGEN$_{I1}$ and DISJCUTGEN$_{I3}$ result in an increase in the number of evaluated nodes.

To evaluate the performance reduction with the DISJCUTGEN procedures further, we compared the time spent in the cut generation procedures in each of the variant. Figure 5.17 compares four implementations of disjunctive cut generation procedure. The time spent in the DISJCUTGEN procedure is mostly the time to solve the CGLPs. Since the CGLP in implementation 1 is very small compared to other implementations, the time spent in DISJCUTGEN$_{I1}$ is negligible. The CGLP in implementation 2 is larger than implementation 1, but much smaller than the CGLPs in implementations 3 and 4 with respect to the number of constraints and variables. The total time spent in the DISJCUTGEN procedure increased significantly in implementation 3 and implementation 4. The number of variables and the main set of constraints in the CGLPs for implementation 3 and implementation 4 are the same. However, in implementation 4, we also added some restrictions for the coefficients of variables in the generated cut, which made the CGLP more difficult to solve and resulted in a more significant increase in time.

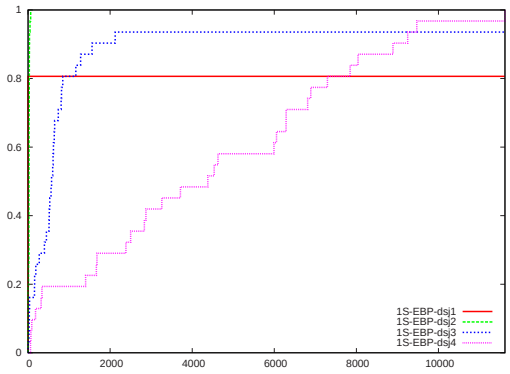


Figure 5.17: Performance profile for the cut generation time with DISJCUTGEN$_{I1}$, DISJCUTGEN$_{I2}$, DISJCUTGEN$_{I3}$, DISJCUTGEN$_{I4}$

The presented results showed that the DISJCUTGEN procedure degrades the algorithm performance. Especially, the cut generation procedures for implementation 3 and implementation 4 are significantly inefficient. Implementation 1 and implementation 2 seem the most promising ones with respect to time. The performance of the variants using implementation 1 and implementation 2 are closest to the algorithm without any cut generation. However, even though the total execution time for the algorithm increases, are the disjunctive valid inequalities generated in our algorithm helping somehow by improving the LP relaxation bound at the related nodes? To be able answer this question, we looked at the optimal LP solutions before and after adding the cuts at each node in each implementation. Let $\bar{x}$ be the fractional solution to the LP relaxation of P^i. First, we evaluate implementation 3 and implementation 4.

In *implementation 3*, almost none of the inequalities generated is violated by $\bar{x}$. The reason is that the cut coefficients are obtained by combining the coefficients found by solving the CGLP. As explained in Section 4.4, for implementation 3, $\bar{x}$ can be mapped to a set of variables (say Y) associated with the arcs in the problem. Recall that we define a one-to-one mapping in order to represent the set Y with a single member (say $\bar{y}$). A disjunctive valid inequality violated by $\bar{y}$ is found first by solving the CGLP. We combine some coefficients considering the other members of Y and then rewrite the obtained inequality in the variable space of P^i. Therefore, the inequality

with combined coefficients is not necessarily violated by $\bar{y}$ and so the corresponding reformulation of the inequality may not be violated by $\bar{x}$. Hence, implementation 3 performs very poorly with respect to LP bound improvement. Therefore, we can easily say that DISJCUTGEN$_{I3}$ is not a promising cut generation procedure.

Implementation 4 was designed to overcome the problems implementation 3 had with generating cuts that are not violated by $\bar{y}$. Instead of combining the coefficients after the cut is generated, implementation 4 adds new conditions to the CGLP to eliminate the combination procedure. Therefore, the generated cut must be violated by $\bar{y}$. Implementation 4 is successful for small instances. However, our computational tests with the 25-customer instances showed that the addition of new conditions to the CGLP increases the solution time significantly. In addition to this drawback, by investigating the solution to (RMP$_k^i$) before and after adding the inequality, we found that there exists some nodes in which even though the generated inequality improves the LP relaxation value at first, the additional column generation steps move the LP solution to a point which has an objective value that is the same or only slightly better than the previous objective value. As seen from the form of the generated valid inequality (4.55), there are coefficients corresponding to the directed arcs in the problem. Because the distance matrix is symmetric, the column generation algorithm generates columns that include different orientations of routes than those that are already a part of the columns in the previous LP solution. The new LP solution satisfies the inequality, but has the same objective value as before because of these new equivalent columns. Therefore, the LP solution value does not change while the set of nonzero pairing variables changes. We provide an example to illustrate this problem.

Example The branch-cut-and-price algorithm with cuts from DISJCUTGEN$_{I4}$ was run for instance p01-f25-v1t1. In one of the nodes, the LP relaxation value before the cut was equal to the LP relaxation value after the cut was added and additional column generation was done. We compared two LP solutions and only listed the non zero variables which were different in these two solutions.

Before cut is added:

$$\bar{z}_1 = 0.188 \quad : \quad \mathrm{f}_0 - 4 - 11 - \mathrm{f}_0 - 22 - 6 - 23 - \mathrm{f}_0 - 13 - 24 - \mathrm{f}_0 - 17 - \mathrm{f}_0$$

$$\bar{z}_2 = 0.16 \quad : \quad \mathrm{f}_0 - 3 - 16 - 18 - 12 - \mathrm{f}_0 - 22 - 6 - 5 - \mathrm{f}_0 - 13 - 24 - \mathrm{f}_0$$

$$\bar{z}_3 = 0.028 \quad : \quad \mathrm{f}_0 - 22 - 6 - 23 - \mathrm{f}_0 - 3 - 18 - 12 - \mathrm{f}_0 - 13 - 5 - \mathrm{f}_0$$

After cut is added and column generation is done:

$$\bar{z}_2 = 0.131 \quad : \quad \mathrm{f}_0 - 3 - 16 - 18 - 12 - \mathrm{f}_0 - 22 - 6 - 5 - \mathrm{f}_0 - 13 - 24 - \mathrm{f}_0$$

$$\bar{z}_4 = 0.188 \quad : \quad \mathrm{f}_0 - 11 - 4 - \mathrm{f}_0 - 23 - 6 - 22 - \mathrm{f}_0 - 24 - 13 - \mathrm{f}_0 - 17 - \mathrm{f}_0$$

$$\bar{z}_5 = 0.029 \quad : \quad \mathrm{f}_0 - 12 - 18 - 16 - 3 - \mathrm{f}_0 - 5 - 6 - 22 - \mathrm{f}_0 - 24 - 13 - \mathrm{f}_0$$

$$\bar{z}_6 = 0.028 \quad : \quad \mathrm{f}_0 - 23 - 6 - 22 - \mathrm{f}_0 - 12 - 18 - 3 - \mathrm{f}_0 - 13 - 5 - \mathrm{f}_0$$

After the cut was added, the column generation generated $\bar{z}_4$ and $\bar{z}_6$ and replaced $\bar{z}_1$ and $\bar{z}_3$ with them. As the sequences of customers in the corresponding pairings are investigated, equivalence of $\bar{z}_1$ and $\bar{z}_4$, and $\bar{z}_3$ and $\bar{z}_6$ are obvious. In addition, $\bar{z}_5$ which is equal to $\bar{z}_2$ took nonzero value in the solution.

Even though there are some nodes in which the disjunctive cuts from DISJCUTGEN$_{I4}$ strictly improve the LP solution values, we can conclude that the branch-and-price algorithm without DISJCUTGEN$_{I4}$ is better of than the branch-cut-and-price algorithm with DISJCUTGEN$_{I4}$. The performance of the overall algorithm is reduced with DISJCUTGEN$_{I4}$ because of the cases in which an improvement in the LP solution cannot be obtained with cuts (similar to the example) and the large CGLP solution times in DISJCUTGEN$_{I4}$.

The form of the cuts generated in *implementation 2* is the same as the forms of the cuts in implementations 3 and 4. The implementation provides coefficients corresponding to arcs in the problem. Therefore, we face the same problem in implementation 4, in some of the nodes, the LP relaxation bound does not improve with the addition of the cut. The symmetric nature of the

problem allows the additional column generation to provide columns that are equal to the existing columns.

In *implementation 1*, we obtained improvement in the LP relaxation bounds in almost all cuts. The new LP solution does not include new equivalent columns (as in the example) since the information passed from the cuts is not symmetric in nature. However, we observed that in some of the nodes the LP relaxation bound improvement is very small (less than 1). Therefore, for these nodes, there is almost no benefit of adding the new cuts, but the cost of searching for the cuts and additional column generation time is significant. This explains why the computational results for implementation 1 do not show an overall performance improvement even though the time spent in DISJCUTGEN$_{I1}$ procedure is small. Another observation from this implementation is that, the number of times that the function searching for disjunctive cuts fails may be large. More clearly, even though we have a fractional x^i_{LP}, we may not be able to find a disjunctive cut. The reason is that the cuts are generated based on a reformulation of the integer model considered at node i. The fractional solution of this reformulation obtained from x^i_{LP} may not be an extreme point of the reformulation polytope. It may be an interior point or on a face of the polytope, which results in a failure in the CGLP that aims to find an inequality violated by that solution.

As a summary, implementation 1 seems to be the best approach compared to other implementations of the disjunctive cut generation procedure with respect to the total time spent in cut generation, total execution time and the LP bound improvement. However, the algorithm with DISJCUTGEN$_{I1}$ terminates abnormally because of the constraint matrix size restriction in more than 20% of the instances while this percentage is at most five for the other variants of the algorithm.

5.6 Value of Integrating Scheduling Decision

In this thesis, we focused on the integrated location, routing and scheduling problem (LRSP). The difference between the LRSP and the integrated location and routing problem (LRP) is the

incorporation of the scheduling decision into the optimization model. The LRP models assume that each vehicle covers at most one route. However, the LRSP models remove this classical assumption and consider the possibility of assigning multiple routes to a single vehicle. At the beginning of the thesis, in Section 1, we claimed that optimizing scheduling decision with the facility location and vehicle routing decisions results in more efficient system designs, and more accurate and cost-effective solutions. We also explained the motivating factors for this statement. In this section, our aim is to empirically show the effect of including scheduling decision into the model by comparing an LRSP model against an LRP model. One can always find the optimal assignment of routes in the solution to vehicles based on the time limit after solving an LRP. We experimentally show the effect of solving an LSRP as opposed to solving an LRP and a scheduling problem sequentially. We focus on the effect of the vehicle capacity.

Models to be Compared. We compare an LRSP with capacity constraints with an LRP with capacity constraints. In the LRP, to be able to satisfy the time limit constraint for each route, we use an LRP with distance constraints (LRPDC). After solving an LRPDC, a scheduling problem for the generated routes can be solved to minimize the number of vehicles. By solving a bin packing problem (BPP), the minimum number of vehicles required to cover all routes, and assignment of routes to vehicles can be found. The capacity of each bin is equal to the time limit, and the length of each route is the space that is allocated by the route in a bin that corresponds to a vehicle. For each facility $j \in J$, we solve the BPP using the following integer programming model.

$$\text{minimize} \quad \sum_{i \in K} y_i \tag{5.2}$$

$$\text{subject to} \quad \sum_{r \in R} t_r x_{ri} \leq L^T y_i \quad \forall i \in K, \tag{5.3}$$

$$\sum_{i \in K} x_{ri} = 1 \quad \forall r \in R, \tag{5.4}$$

$$x_{ri} \in \{0,1\} \quad \forall r \in R, i \in K, \tag{5.5}$$

$$y_i \in \{0,1\} \quad \forall i \in K, \tag{5.6}$$

where R is the set of routes for facility j, K is the set of vehicles and t_r is the trip time for route r. y_i is equal to 1 if the vehicle i is used, and x_{ri} is 1 if route r is assigned to vehicle i for all $r \in R$ and $i \in K$.

To solve the LRPDC, we use a set partitioning-based formulation which has the same structure as the formulation for the LRSP with the only difference being the definition of their column sets. In the LRPDC, a column that is generated using the column generation algorithm represents a single feasible vehicle route satisfying the vehicle capacity and the time limit constraints. In the LRPDC, the pricing problem is an ESPPRC, where the vehicle capacity and the time limit are the resources. To solve the pricing problem in the LRPDC, we use only the first phase of the 2p-ESPPRC (described in Section 3.2.2). By using the first phase of the 2p-ESPPRC-LL and 2p-ESPPRC-CS, we can obtain heuristic pricing algorithms for the LRPDC. In Chapter 6, we describe a branch-and-price algorithm to solve an LRP in detail.

Implementation Details and Discussion of the Results. In our experiments, we used the 25-customer instances with smaller time limit value (instances labeled with "t1"). Recall that two different vehicle capacities calculated with $\alpha = 0.1$ and 0.2 were used in these instances. These α values were specifically chosen to be small for the LRSP in order to result in smaller vehicle capacity values requiring multiple trips to a facility. We expect to see that as the vehicle capacity gets larger, the benefit of solving the LRSP instead of the LRPDC is reduced. Therefore, we populated the set of instances with two more capacity values referred to as L_3^V and L_4^V that were calculated by setting $\alpha = 0.5$ and $\alpha = 0.8$. We indicate these capacity values using "v3" and "v4" in the labels of the instances. Table 5.12 lists all vehicle capacity values for each instance group. We obtained 32 instances for this experiment. For each model (LRSP and LRPDC) and instance, we used the one-stage branch-and-price algorithm with E-COLGEN(0,0,50,350) with a time limit of 8 CPU hours.

Figure 5.18 compares the total solution time for each problem. Figure indicates that the branch-and-price algorithm for the LRDC performs better than the branch-and-price algorithm

Table 5.12: Vehicle capacity values

Ref	L_1^V ($\alpha = 0.1$)	L_2^V ($\alpha = 0.2$)	L_3^V ($\alpha = 0.5$)	L_4^V ($\alpha = 0.8$)
p01	50	70	120	170
p03	60	80	140	190
p07	50	70	110	150

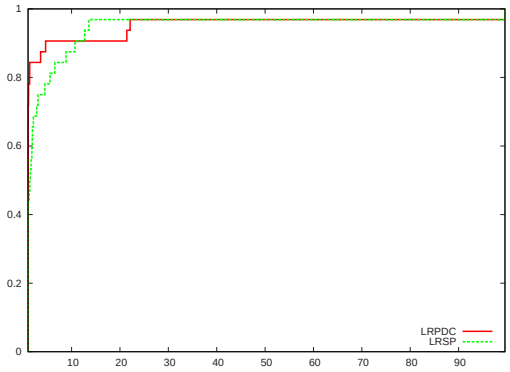


Figure 5.18: Performance profile comparing the execution time of 1S-EBP for LRSP and LRPDC

for the LRSP with respect to computational time. This is expected because the pricing problem used to solve the LRSP is more difficult than the pricing problem used in the solution of the LRPDC. For the set of instances that were used in the experiment, we calculated that the column generation time in the LRSP was about 2.9 times larger on average than the column generation time in the LRPDC.

In Tables 5.13 and 5.14, we report the solution details for each instance and for each model, including indices of facilities that are open in the solution (Fac. Idx), number of vehicles (# vec.), total facility fixed cost (Fac. Cost), total operating cost (Route Cost), and total cost (TCost). The fourth column in Table 5.13 (# vec. w. BPP) reports the number of vehicles obtained by solving the BPP for the LRPDC solution to reduce the number of vehicles and the seventh column (TCost w. BPP) for the LRPDC reports the total cost that was calculated with the reduced number of vehicles after the BPP.

First, we compared the optimal total costs of the problems reported in columns labeled with

Table 5.13: Solution details for LRPDC + BPP

Data File	Fac. Idx	# vec.	# vec. w. BPP	Fac. Cost	Route Cost	TCost w. BPPt
p01-f25-v1t1	0,4	9	4	3368	447	4715
p01-f25-v2t1	0,4	7	4	3368	377	4645
p01-f25-v3t1	0,4	4	4	3368	307	4575
p01-f25-v4t1	0,4	4	4	3368	299	4567
p01-l25-v1t1	0,4	8	5	3368	547	5040
p01-l25-v2t1	0,4	6	5	3368	472	4965
p01-l25-v3t1	0,4	3	3	3368	366	4409
p01-l25-v4t1	0,4	3	3	3368	354	4397
p03-f25-v1t1	0,2	10	5	3396	541	5062
p03-f25-v2t1	0,4	8	5	3368	474	4967
p03-f25-v3t1	0,4	4	4	3368	363	4631
p03-f25-v4t1	0,4	4	4	3368	361	4629
p03-m25-v1t1	0,4	9	5	3368	524	5017
p03-m25-v2t1	0,4	7	4	3368	421	4689
p03-m25-v3t1	0,4	6	4	3368	336	4604
p03-m25-v4t1	0,4	4	4	3368	368	4636
p03-l25-v1t1	0,4	7	6	3368	535	5253
p03-l25-v2t1	0,4	5	5	3368	491	4984
p03-l25-v3t1	0,4	4	4	3368	396	4664
p03-l25-v4t1	0,4	3	3	3368	395	4438
p07-f25-v1t1	0,2	7	4	3396	465	4761
p07-f25-v2t1	0,2	5	4	3396	412	4708
p07-f25-v3t1	0,4	4	3	3368	362	4405
p07-f25-v4t1	0,4	3	3	3368	354	4397
p07-s25-v1t1	0,4	8	7	3368	647	5590
p07-s25-v2t1	0,4	6	5	3368	513	5006
p07-s25-v3t1	0,4	4	4	3368	437	4705
p07-s25-v4t1	0,4	3	3	3368	397	4440
p07-t25-v1t1	0,2	8	5	3396	573	5094
p07-t25-v2t1	0,2	6	5	3396	457	4978
p07-t25-v3t1	0,4	4	4	3368	407	4675
p07-t25-v4t1	0,4	3	3	3368	357	4400
average		5.5	4.22	3372.38	429.84	4751.44

Table 5.14: Solution details for LRSP

Data File	Fac. Idx	# vec.	Fac. Cost	Route Cost	TCost
p01-f25-v1t1	0,4	4	3368	431	4699
p01-f25-v2t1	0,4	3	3368	382	4425
p01-f25-v3t1	0,4	3	3368	307	4350
p01-f25-v4t1	0,4	3	3368	300	4343
p01-l25-v1t1	0,1	4	3460	539	4899
p01-l25-v2t1	0,4	4	3368	491	4759
p01-l25-v3t1	0,4	3	3368	366	4409
p01-l25-v4t1	0,4	3	3368	354	4397
p03-f25-v1t1	0,2	5	3396	541	5062
p03-f25-v2t1	0,4	4	3368	490	4758
p03-f25-v3t1	0,4	4	3368	363	4631
p03-f25-v4t1	0,4	3	3368	366	4409
p03-m25-v1t1	0,4	4	3368	517	4785
p03-m25-v2t1	0,4	4	3368	421	4689
p03-m25-v3t1	0,4	3	3368	361	4404
p03-m25-v4t1	0,4	3	3368	339	4382
p03-l25-v1t1	0,2	4	3396	546	4842
p03-l25-v2t1	0,4	4	3368	473	4741
p03-l25-v3t1	0,4	3	3368	408	4451
p03-l25-v4t1	0,4	3	3368	395	4438
p07-f25-v1t1	0,2	4	3396	465	4761
p07-f25-v2t1	0,2	4	3396	407	4703
p07-f25-v3t1	0,4	3	3368	362	4405
p07-f25-v4t1	0,4	3	3368	354	4397
p07-s25-v1t1	0,4	5	3368	605	5098
p07-s25-v2t1	0,4	4	3368	519	4787
p07-s25-v3t1	0,4	4	3368	427	4695
p07-s25-v4t1	0,4	3	3368	397	4440
p07-t25-v1t1	0,1	4	3460	538	4898
p07-t25-v2t1	0,2	4	3396	471	4767
p07-t25-v3t1	0,4	3	3368	407	4450
p07-t25-v4t1	0,4	3	3368	357	4400
average		3.59	3378.13	428.09	4614.81

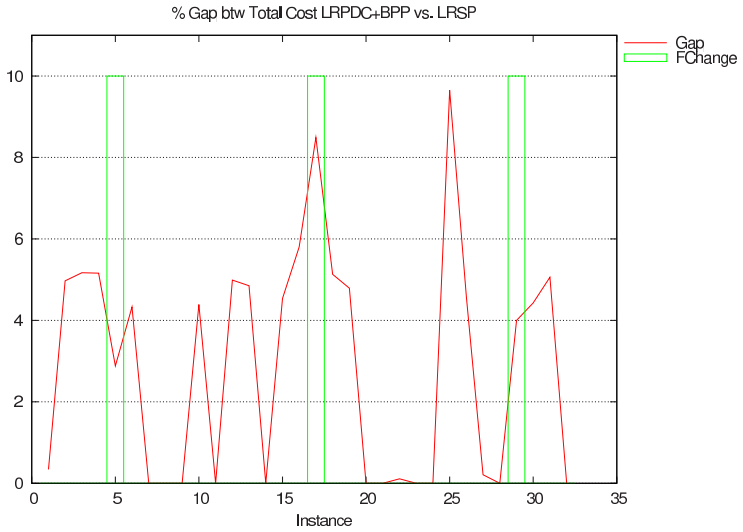


Figure 5.19: % Improvement in the total cost with LRSP compared to LRPDC + BPP

TCost in Tables 5.13 and 5.14. The total cost in the LRPDC with the BPP is generally larger than the cost obtained by solving the LRSP, with eight instances, having equal costs. For each instance, we calculated the % cost improvement obtained by solving an LRSP. Figure 5.19 shows the % improvement calculated for each instance. On average, the total cost obtained by solving an LRSP is 2.9% better than the cost obtained by solving the LRPDC and the BPP sequentially. Figure 5.19 also indicates the instances in which the set of open facilities in the LRPDC solution differs from the set in the LRSP solution using bars in such cases. As it can be seen, in three instances, the set of open facilities is different. This shows that integration of the scheduling decision may effect the facility location decision. In Tables 5.13 and 5.14, we can observe that the routing costs in the LRPDC and the LRSP may also differ. In eight instances, the routing cost in the LRPDC is smaller than the routing cost in the LRSP solution. Since the weight of the routing cost is small with respect to facility and vehicle fixed costs, the major cost difference between these two problems is because of the difference in the number of vehicles.

Figure 5.20 compares the number of vehicles in the LRPDC solution before and after solving the resulting BPP with the number of vehicles in the LRSP solution. In all but seven instances,

Table 5.15: Average number of vehicles for each vehicle capacity value

Capacity	average # of vehicles				cost saving /year
	LRPDC	LRP+BPP	LRSP	Diff	
L_1^V	8.25	5.13	4.25	0.88	51187
L_2^V	6.25	4.63	3.88	0.75	43875
L_3^V	4.13	3.75	3.25	0.5	29250
L_4^V	3.38	3.38	3	0.38	21937.5
Average	5.5	4.22	3.59	0.63	36562.5

the number of vehicles in the LRPDC solution before solving the resulting BPP is larger than the number of vehicles in the LRSP solution. In all of the instances with vehicle capacity "v1" or "v2", the number of vehicles in the LRPDC is strictly larger than the number of vehicles in the LRSP. As the vehicle capacity gets larger, the number of vehicles in the LRPDC gets closer to the number of vehicles in the LRSP.

Table 5.15 compares and summarizes the average number of vehicles as well as the cost saving for each vehicle capacity and for each model. On average, the number of vehicles in the LRPDC is 5.5, while this average is 3.59 in the LRSP. Solving the BPP after the LRPDC reduces the number of vehicles that are needed. The BPP is able to reduce the number of vehicles when vehicle capacities are smaller ("v1" and "v2"). As the vehicle capacity gets larger, solving the BPP does not improve the number of vehicles used. For instance, solving the BPP is not able to reduce the number of vehicles in any of the instances that are labeled with "v4", only in two of the instances that are labeled with "v3", the number of vehicles is improved. On average, the number of vehicles after the BPP is 4.22, which is still larger than the average number of vehicles in the LRSP. This difference results in about $142 vehicle cost difference per day with a vehicle fixed cost of $225. Thus, on average, solving an LRSP results in a reduction of about $36600 in the total vehicle fixed cost per year (based on 260 days). When we only consider instances with small vehicle capacities such as instances that are labeled with "v1" and "v2", on average, the difference between the number of vehicles in the LRPS and the number of vehicles in the LRPDC and the BPP is 0.82, which results in more than $47000 savings per year.

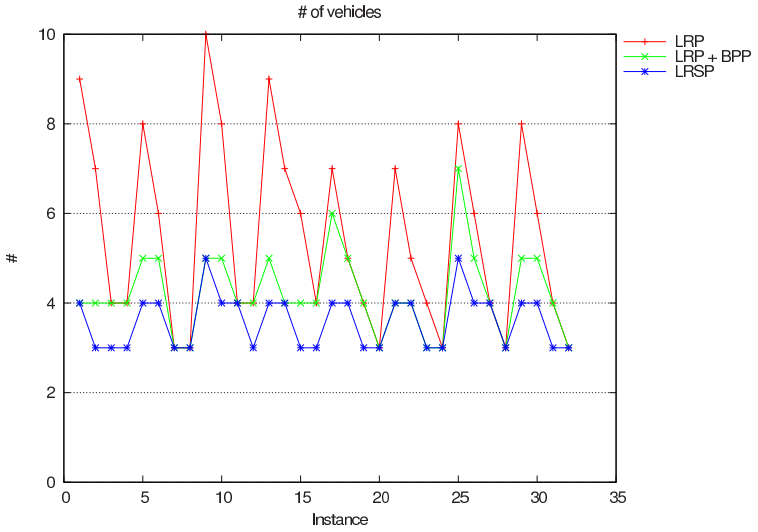


Figure 5.20: # of vehicles with LRPDC, LRPD + BPP and LRSP

These experiments computationally demonstrate the difference between solving an LRSP and an LRPDC. The results indicate the benefit of integrating scheduling decision with the location and routing decisions. This action may even change the facility locations that are selected. The number of vehicles in the LRSP and the LRPDC differ significantly, especially with small vehicle capacities. Even if we solve a BPP after solving an LRPDC, the number of vehicles may be larger than that in the LRSP solution. Thus, we can conclude that solving an LRSP is more preferable than using an LRPDC model when the vehicle fixed cost are large. The instances that are used in the experiments include only 25 customers. The number of vehicles used, as well as the total cost, may be improved even more substantially in larger instances. The difference between the routing costs are not significant. Therefore, in the absence of the vehicle fixed costs or in the absence of a restriction on the number of vehicles, the difference between solving an LRPDC and an LRSP may not be significant.

5.7 Comparing Integrated and Sequential Optimization

In this section, we aim to empirically compare the integrated optimization (LRSP) and the optimization of facility location, vehicle routing and vehicle scheduling decisions sequentially, which we refer to as sequential optimization.

Models to be Compared. In sequential optimization, we first solve a capacitated facility location problem in which each customer can only be assigned to a single facility (CFLPSS). We solve the following optimization problem:

$$
\begin{aligned}
\text{Min} \quad & \sum_{j \in J} C_j^F t_j + C^O \sum_{j \in J} \sum_{i \in I} (T_{ji} + T_{ij}) v_{ji} & \\
\text{s.t.} \quad & \sum_{j \in J} v_{ji} = 1 & \forall i \in I, \\
& \sum_{i \in I} D_i v_{ji} - L_j^F t_j \leq 0 & \forall j \in J, \\
& t_j \in \{0,1\}, v_{ji} \in \{0,1\} & \forall j \in J, i \in I,
\end{aligned}
$$

where $v_{ji} = 1$ if customer i is assigned to facility j for all $i \in I$, $j \in J$. Then, we solve the capacitated LRPDC problem in which only the facilities that are open in the solution of CFLPSS are set to be open. Finally, as in the previous section, one BPP is solved for each open facility to optimize the scheduling of vehicles in the solution to LRPDC. Comparing the solution to the LRSP with the solution to the sequential optimization would evaluate the value of optimizing facility location, vehicle routing and vehicle scheduling decisions interdependently.

Implementation Details and Discussion of the Results. In our experiments, we used the 32 25-customer instances that are labeled with "v1t1", "v1t2", "v2t1" and "v2t2". For each approach (LRSP and sequential optimization) and instance, we used the one-stage branch-and-price algorithm with E-COLGEN(0,0,50,350) with a time limit of 6 CPU hours. In sequential optimization, the branch-and-price was used to solve the LRPDC; CFLPSS and BPP were solved using CPLEX

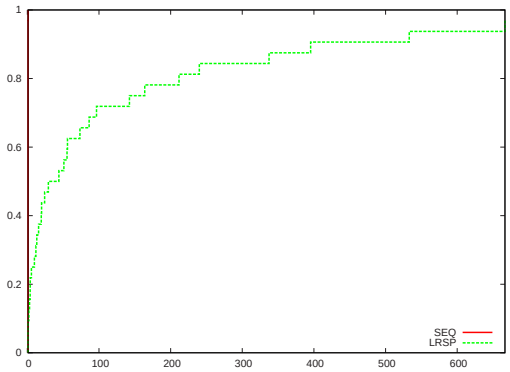


Figure 5.21: Performance profile comparing the execution time of LRSP and Sequential optimization

9.1. Figure 5.21 compares the total solution time for each approach. Figure indicates that total time in the sequential approach is much better than the total time required to solve the LRSP.

In Tables 5.13 and 5.14, we report the solution details for each instance and for each solution approach, including indices of facilities that are open in the solution (Fac. Idx), number of vehicles (# vec.), total facility fixed cost (Fac. Cost), total operating cost (Route Cost), and total cost (TCost). The total cost obtained using the sequential approach is generally larger than the cost obtained by solving the LRSP, with five out of 32 instances, having equal costs. For each instance, we calculated the % cost improvement obtained by solving an LRSP and visualize these values in Figure 5.22. On average, the total cost obtained by solving an LRSP is 3.8% better than the cost obtained from the sequential optimization. Average total cost saving per year is about $47000. Figure 5.22 also indicates the instances in which the sets of open facilities are different for two solution approaches. In 18 out of 32 instances, the sets of open facilities are different. This shows that the integrated optimization decision may effect the facility location decision. In Figure 5.23, we compare the vehicle fixed costs and routing costs for the LRSP and the sequential optimization. Figure 5.23(a) presents the % improvement in total vehicle fixed cost obtained by solving an LRSP as opposed to applying the sequential optimization for each instance. On average, the integrated optimization results in %21 improvement in the total vehicle fixed cost which is equal to $44000

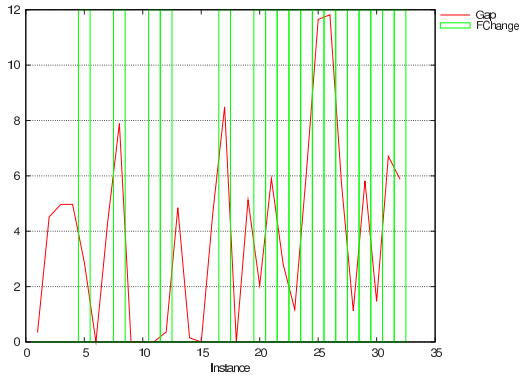
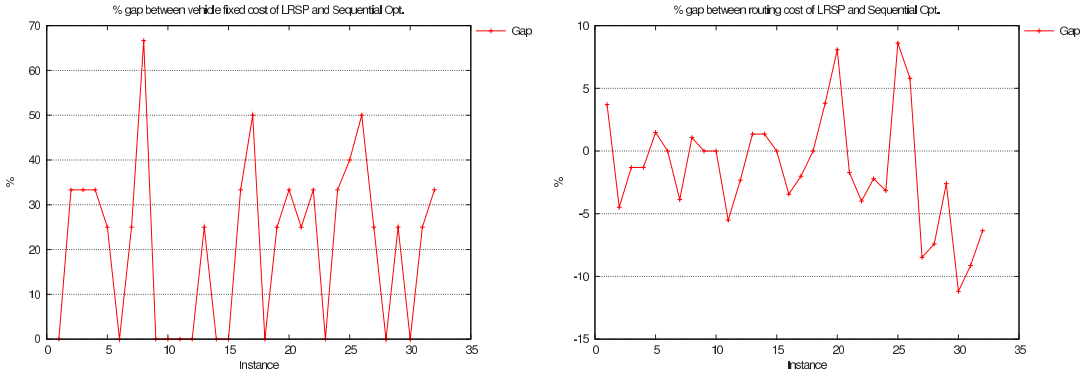


Figure 5.22: % Improvement in the total cost with LRSP compared to sequential optimization



(a) % Improvement in the vehicle fixed cost

(b) % Improvement in the routing cost

Figure 5.23: LRSP versus sequential optimization

saving per year. Figure 5.23(b) presents the % improvement in total routing cost. In 18 out of 32 instances, sequential optimization approach results in a lower routing cost than the LRSP. On average, the routing cost is %1.4 higher in the LRSP.

These experiments computationally demonstrate the value of solving an LRSP instead of optimizing each decision sequentially. In conclusion, solution to an LRSP may result in large cost saving.

Table 5.16: Solution details for Sequential Optimization

Data File	Fac. Idx	# vec.	Fac. Cost	Route Cost	TCost
p01-f25-v1t1	0,4	4	3368	447	4715
p01-f25-v1t2	0,4	4	3368	447	4715
p01-f25-v2t1	0,4	4	3368	377	4645
p01-f25-v2t2	0,4	4	3368	377	4645
p01-l25-v1t1	0,4	5	3368	547	5040
p01-l25-v1t2	0,4	4	3368	547	4815
p01-l25-v2t1	0,4	5	3368	472	4965
p01-l25-v2t2	0,4	5	3368	472	4965
p03-f25-v1t1	0,2	5	3396	541	5062
p03-f25-v1t2	0,2	4	3396	541	4837
p03-f25-v2t1	0,2	4	3396	463	4759
p03-f25-v2t2	0,2	4	3396	463	4759
p03-m25-v1t1	0,4	5	3368	524	5017
p03-m25-v1t2	0,4	4	3368	524	4792
p03-m25-v2t1	0,4	4	3368	421	4689
p03-m25-v2t2	0,4	4	3368	421	4689
p03-l25-v1t1	0,4	6	3368	535	5253
p03-l25-v1t2	0,4	4	3368	535	4803
p03-l25-v2t1	0,4	5	3368	491	4984
p03-l25-v2t2	0,4	4	3368	482	4750
p07-f25-v1t1	0,1	5	3460	457	5042
p07-f25-v1t2	0,1	4	3460	457	4817
p07-f25-v2t1	0,1	4	3460	398	4758
p07-f25-v2t2	0,1	4	3460	398	4758
p07-s25-v1t1	0,1	7	3460	657	5692
p07-s25-v1t2	0,1	6	3460	657	5467
p07-s25-v2t1	0,1	5	3460	475	5060
p07-s25-v2t2	0,1	4	3460	475	4835
p07-t25-v1t1	2,4	5	3534	524	5183
p07-t25-v1t2	2,4	4	3534	524	4958
p07-t25-v2t1	2,4	5	3534	428	5087
p07-t25-v2t2	2,4	4	3534	428	4862
Average		4.53	3415.25	484.53	4919.31

Table 5.17: Solution details for LRSP

Data File	Fac. Idx	# vec.	Fac. Cost	Route Cost	TCost
p01-f25-v1t1	0,4	4	3368	431	4699
p01-f25-v1t2	0,4	3	3368	468	4511
p01-f25-v2t1	0,4	3	3368	382	4425
p01-f25-v2t2	0,4	3	3368	382	4425
p01-l25-v1t1	0,1	4	3460	539	4899
p01-l25-v1t2	0,4	4	3368	547	4815
p01-l25-v2t1	0,4	4	3368	491	4759
p01-l25-v2t2	0,1	3	3460	467	4602
p03-f25-v1t1	0,2	5	3396	541	5062
p03-f25-v1t2	0,2	4	3396	541	4837
p03-f25-v2t1	0,4	4	3368	490	4758
p03-f25-v2t2	0,4	4	3368	474	4742
p03-m25-v1t1	0,4	4	3368	517	4785
p03-m25-v1t2	0,4	4	3368	517	4785
p03-m25-v2t1	0,4	4	3368	421	4689
p03-m25-v2t2	0,4	3	3368	436	4479
p03-l25-v1t1	0,2	4	3396	546	4842
p03-l25-v1t2	0,4	4	3368	535	4803
p03-l25-v2t1	0,4	4	3368	473	4741
p03-l25-v2t2	2,4	3	3534	446	4655
p07-f25-v1t1	0,2	4	3396	465	4761
p07-f25-v1t2	2,4	3	3534	476	4685
p07-f25-v2t1	0,2	4	3396	407	4703
p07-f25-v2t2	0,2	3	3396	411	4482
p07-s25-v1t1	0,4	5	3368	605	5098
p07-s25-v1t2	0,4	4	3368	621	4889
p07-s25-v2t1	0,4	4	3368	519	4787
p07-s25-v2t2	0,4	4	3368	513	4781
p07-t25-v1t1	0,1	4	3460	538	4898
p07-t25-v1t2	0,2	4	3396	590	4886
p07-t25-v2t1	0,2	4	3396	471	4767
p07-t25-v2t2	0,1	3	3460	457	4592
Average		3.78	3396.88	491.16	4738.81

Chapter 6

Location and Routing Problems

In this chapter, we investigate the location and routing problem (LRP). Given a set of candidate facility locations and a set of customer locations, the objective of the LRP is to determine the number and location of facilities and construct a set of vehicle routes from facilities to customers in such a way as to minimize total system cost. The system cost may include both the fixed and operating costs of both facilities and vehicles. In this study, we consider an LRP with capacity constraints on both the facilities and the vehicles.

As mentioned in Chapter 1, the LRP is a special case of the LRSP. To see this, consider an LRSP instance in which there is one vehicle available at each facility and the time limit for the vehicle is infinite. In addition, we set the fixed cost of a vehicle to zero, but the arc costs from facilities to customers are modified by adding a vehicle fixed cost, if there is one. The solution of the described LRSP instance gives a solution to the corresponding LRP instance.

Few papers have been devoted to combined location and routing models, and even fewer papers have been devoted to the study of exact algorithms for the LRP. The literature related to the LRP was reviewed in Section 1.3.3. In this chapter, we modify the branch-and-price algorithm that is developed for the LRSP to solve the LRP under capacity restrictions, and report on our computational experience solving both randomly generated instances and instances from the literature.

The remainder of the chapter is organized as follows. Section 6.1 presents a formal description of the problem and provides two formulations. Section 6.2 describes the heuristic and the exact algorithms for the set partitioning formulation. Section 6.3 provides some computational results evaluating the performance of the algorithms.

6.1 Problem Definition and Formulations

The objective of the LRP is to select a subset of facilities and construct an associated set of vehicle routes serving customers at minimum total cost, where the cost includes the fixed and operating costs of both facilities and vehicles. The constraints of the problem are as follows:

1. Each customer must be serviced by exactly one vehicle.
2. Each vehicle must be assigned to exactly one facility at which it must start and end its route.
3. The total demand of the customers assigned to a route must not exceed the vehicle capacity.
4. The total demand of the customers assigned to a facility must not exceed the capacity of the facility.

In the literature, most of the exact methods developed for the described LRP or its special cases are based on the two-index vehicle flow formulation of the problem. To the best of our knowledge, an exact solution algorithm based on a set partitioning formulation has not been applied to the case of the LRP with capacity constraints. Next, we present a set partitioning formulation of the problem.

6.1.1 Set Partitioning Formulation

Here, we utilize a modified version of the set partitioning formulation for the LRSP (SPP-LRSP) that is presented in Chapter 2. We define the set R to be a set indexing all vehicle routes feasible with respect to vehicle capacity and originating from and returning to the same facility. We let

$R_j \subseteq R$ index routes associated with vehicles assigned to facility j for all $j \in J$. We use the same notation that is presented in Section 2.1 to represent the facility fixed costs (C^F), the facility capacities (C^F) and the demand (D).

The definitions of some parameters and variables that are used in (SPP-LRSP) change slightly depending on the column index set R. We redefine the following parameters and variables for the LRP formulation.

Parameters:

$$C_r = \text{operating and fixed cost of route } r \in R,$$

$$a_{ir} = \begin{cases} 1 & \text{if customer } i \in I \text{ is assigned to route } r \in R, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Decision Variables:

$$z_r = \begin{cases} 1 & \text{if route } r \in R \text{ is selected, and} \\ 0 & \text{otherwise,} \end{cases}$$

$$t_j = \begin{cases} 1 & \text{if facility } j \in J \text{ is selected to be open, and} \\ 0 & \text{otherwise,} \end{cases}$$

The formulation referred to as (SPP-LRP) is as follows.

$$\text{Min} \quad \sum_{j \in J} C_j^F t_j + \sum_{r \in R} C_r z_r \tag{6.1}$$

(SPP-LRP) s. t.

$$\sum_{r \in R} a_{ir} z_r = 1 \qquad \forall i \in I, \tag{6.2}$$

$$\sum_{r \in R_j} \sum_{i \in I} D_i a_{ir} z_r - C_j^F t_j \leq 0 \qquad \forall j \in J, \tag{6.3}$$

$$z_r \in \{0,1\} \quad \forall r \in R, \tag{6.4}$$

$$t_j \in \{0,1\} \quad \forall j \in J. \tag{6.5}$$

The objective function (6.1) seeks to minimize the total cost, which includes the fixed cost of the selected facilities and the fixed and operating costs of the vehicles. Constraints (6.2) guarantee that each customer location is served by exactly one route. Constraints (6.3) ensure that the total demand of the selected routes for a facility does not exceed the facility capacity. Constraints (6.4) and (6.5) are standard binary restrictions on the variables.

6.1.2 Vehicle Flow Formulation

Here, we present a three-index vehicle flow formulation for the LRP similar to the model developed for the LRSP in Section 2.2.1 (G-LRSP). We use the same notation that is presented in Section 2.1 and in addition, we define a variable u_{ih} that takes value 1 if vehicle h visits customer i and 0, otherwise, $\forall i \in I, h \in H$. A vehicle flow formulation of the LRP that we refer to as (G-LRP) is as follows:

$$\text{Min} \quad \sum_{j\in J} C_j^F t_j + C^O \sum_{h\in H} \sum_{(i,k)\in A} T_{ik} x_{ikh} \tag{6.6}$$

(G-LRP) s.t.

$$\sum_{h\in H} \sum_{k\in N} x_{ikh} = 1 \qquad \forall i \in I, \tag{6.7}$$

$$\sum_{k\in N} x_{ikh} - \sum_{k\in N} x_{kih} = 0 \qquad \forall i \in N, h \in H, \tag{6.8}$$

$$\sum_{k\in N} x_{kih} - u_{ih} = 0 \qquad \forall i \in I, h \in H, \tag{6.9}$$

$$u_{ih} - \sum_{k\in S} \sum_{l\in V\setminus S} x_{klh} \leq 0 \qquad \forall S \subseteq I, i \in S, h \in H, \tag{6.10}$$

$$\sum_{i\in I} D_i u_{ih} - C^V \leq 0 \qquad \forall h \in H, \tag{6.11}$$

$$\sum_{h\in H_j} \sum_{i\in I} D_i u_{ih} - C_j^F t_j \leq 0 \qquad \forall j \in J, \tag{6.12}$$

$$x_{ikh} \in \{0,1\} \quad \forall i,k \in V, h \in H, \tag{6.13}$$

$$u_{ih} \in \{0,1\} \quad \forall i \in I, h \in H, \tag{6.14}$$

$$t_j \in \{0,1\} \quad \forall j \in J. \tag{6.15}$$

In (G-LRP), the objective function (6.6) seeks to minimize the total cost, which includes the fixed cost of the selected facilities and the operating cost of the vehicles. Vehicle fixed costs can easily be incorporated into the model by increasing the cost of traveling from the facility to each customer location by a fixed amount. Constraints (6.7) specify that exactly one vehicle must service customer i. Constraints (6.8) require that each vehicle should enter and leave a location the same number of times. Constraints (6.9) determine the assignment of customers to vehicles. Constraints (6.10) eliminate subtours, i.e., routes that do not include a facility. Constraints (6.11) are vehicle capacity constraints. Constraints (6.12) ensure that the capacity of a facility is not exceeded by demand flows to customer locations. For notational convenience, we assume that variables associated with travel between different facilities or travel between a customer and a facility using a truck not associated with that facility are fixed to zero. Constraints (6.13), (6.14), and (6.15) are the set of binary restrictions on the variables.

We can demonstrate the relationship between (G-LRP) and (SPP-LRP) by applying Dantzig-Wolfe decomposition (DWD) to (G-LRP). This approach will yield a model that is equivalent to (SPP-LRP). In the application of the DWD method, constraints (6.7) and (6.12) form the master problem, while the remaining constraints define the subproblem. This subproblem can then be decomposed by facility and further by vehicle. The equivalence of (SPP-LRP) and the DWD of (G-LRP) follows after a further reduction in the subproblem column set based on symmetry and constraints in the master problem. A similar study has already been done for the LRSP in Section 2.4 and deriving similar results for the LRP is straightforward. Therefore, we skip the details here.

6.2 Solution Algorithm

Comparing (SPP-LRP) and (SPP-LRSP), we see that the only difference between them is the definition of the column set. The columns of (SPP-LRSP) correspond to the set of feasible *pairings* for all facilities. Since the scheduling decision is not considered in an LRP, (SPP-LRP) is defined over the set of feasible *routes* for all facilities (denoted by R). Based on this similarity, we can

easily modify our solution algorithms for the LRSP to solve the LRP. We alter our branch-and-price implementation in order to utilize formulation (SPP-LRP) and generate routes instead of pairings in the column generation part of the algorithm.

6.2.1 Strengthening the Set Partitioning-Based Formulation

As in the LRSP case (described in Section 2.3), we first strengthen (SPP-LRP) by adding the following additional valid inequalities:

$$\sum_{r \in R_j} a_{ir} z_r - t_j \leq 0 \qquad \forall i \in I, j \in J, \tag{6.16}$$

$$\sum_{j \in J} t_j \geq N^F, \tag{6.17}$$

$$\sum_{r \in R_j} z_r = v_j \qquad \forall j \in J, \tag{6.18}$$

$$\sum_{j \in J} v_j \geq \left\lceil \frac{\sum_{i \in I} D_i}{C^V} \right\rceil, \tag{6.19}$$

$$v_j \geq t_j \qquad \forall j \in J, \tag{6.20}$$

$$v_j \in \mathbb{Z}_+ \qquad \forall j \in J, \tag{6.21}$$

where v_j represents the number of vehicles used at facility j, $\forall j \in J$ and N^F is a lower bound on the number of facilities that must be open in any feasible solution, calculated as follows:

$$N^F = \text{argmin}_{\{l=1,\ldots,|J|\}} \left(\sum_{t=1}^{l} C^F_{j_t} \geq \sum_{i \in I} D_i \right) \text{ s.t. } C^F_{j_1} \geq C^F_{j_2} \geq \ldots \geq C^F_{j_n}.$$

Constraints (6.16) force a facility to be open if any customer is assigned to it. Constraint (6.17) sets a lower bound on the total number of facilities required in any integer feasible solution. In the LRSP case, constraints (6.16) and constraint (6.17) are shown computationally to improve the LP relaxation of the model. As in the LRSP, constraints (6.18) are only added to facilitate branching on the integrality of the number of vehicles at each facility in the branch-and-price algorithm.

Constraint (6.19) sets a lower bound for the total number of vehicles in the solution. Finally, constraints (6.20) force the number of vehicles used at a facility to be at least 1 if the facility is open. We refer to formulation (6.1)–(6.5) and (6.16)–(6.21) as (ASPP-LRP) for the rest of the chapter.

6.2.2 Branch-and-Price Algorithm

Since we are modifying the branch-and-price algorithm used for the LRSP, which is described in detail in Chapter 3, we only explain here the modified parts of the algorithm. The main structure of the algorithm BRANCH-PRICE, the COLGEN procedure, the branching rules, and the versions of the branch-and-price algorithms designed are the same as for the LRSP case. The formulation (ASPP-LRP) contains a variable for each possible vehicle route originating from each facility. Therefore, we alter the solution algorithms for the pricing problem for the LRSP in order to generate feasible routes instead of pairings.

Pricing Problem. The pricing problem (for each facility) for the LRP is also an ESPPRC. Consider a network with $|I| + 2$ nodes, one node for each customer, one for the facility j as a source node and a copy of the facility j as a sink node. We assign each customer node a demand equal to its demand in the original problem and let the cost of each arc (i, l) in the network be equal to the contribution of arc (i, l) to the reduced cost of a column that represents a route including arc (i, l). The reduced cost function for the RMP in the LRP is similar to the reduced cost function in the LRSP case. A shortest path from source to sink visiting a customer at most once and satisfying the constraint that the total demand of customers included in the path does not exceed the vehicle capacity then corresponds to a vehicle route. The total cost of the path plus the fixed costs (if any) is the reduced cost of the associated column.

To solve the ESPPRC, we use Feillet et al. (2004)'s label setting algorithm with a single resource (the vehicle capacity) which is described in Section 3.2.2 and referred to as 1p-ESPPRC. Note that the algorithm described in Section 3.2.2 is simplified by removing the time limit from the

resource set in order to use it for the LRP, however, we still use notation 1p-ESPPRC to denote the exact pricing algorithm used in this chapter. To enhance the efficiency of the overall branch-and-price algorithm, we use the two heuristic extensions of the exact pricing algorithm: ESPPRC-LL and ESPPRC-CS presented in Section 3.2.3. We refer to these algorithms as 1p-ESPPRC-LL(l) and 1p-ESPPRC-CS(u).

As in the LRSP case, in the branch-and-price algorithm, we apply pricing algorithms 1p-ESPPRC, 1p-ESPPRC-LL(l) and 1p-ESPPRC-CS(u) in combination in the column generation algorithms to improve the overall performance. Two designs of the column generation algorithm, E-COLGEN(u, U, l, L) (Figure 3.4) and H-COLGEN(u, U, l, L) (Figure 3.5), can be modified for the LRP. In the modified versions, we call 1p-ESPPRC, 1p-ESPPRC-LL(l) and 1p-ESPPRC-CS(u) instead of 2p-ESPPRC, 2p-ESPPRC-LL(l) and 2p-ESPPRC-CS(u) in combination. In this chapter, we still use representations E-COLGEN(u, U, l, L) and H-COLGEN(u, U, l, L) to denote the exact and heuristic column generation algorithms for the LRP.

In addition to the exact and heuristic approaches to solve the pricing problem, we designed an algorithm that solves a relaxation of the pricing problem. We refer to this algorithm as 2-Cyc-SPPRC-PE(C) where C is a subset of customers.

2-Cyc-SPPRC-PE(C). Recall that the shortest path problem with resource constraints (SPPRC) is a relaxation of the ESPPRC in which the path may visit some customers more than once. The SPPRC is solvable in pseudo-polynomial time, but use of this further relaxation of the column generation subproblem results in reduced bounds. Eliminating solutions containing cycles of length two strengthens this relaxation of the pricing problem and improves the bound (for details of the algorithm, see Irnich and Desaulniers (2005)). We refer to this pricing algorithm as 2-Cyc-SPPRC. In addition, we can also generate paths that are elementary with respect to a given subset of customers. We call the resulting algorithm 2-Cyc-SPPRC-PE(C), where C is the set of customers that can be visited at most once in the resulting solutions.

R-COLGEN(m)
Input: An LRP instance, a subset of columns C_G and an integer m ($< |I|$).
Output: A lower bound for the LP relaxation of the problem at the node and a column set modified by adding new columns.

1. Do $P \rightarrow$ 2-Cyc-SPPRC.
2. If $P = \emptyset$, STOP. Otherwise, consider the column with minimum cost in P and choose at most m customers that are visited more than once. Let C_1 be the set of these customers. If $C_1 = \emptyset$ (the path is already elementary), then add the column to set C_G and return to step 1. Otherwise, proceed.
3. Do $P \rightarrow$ 2-Cyc-SPPRC-PE(C_1).
4. If $P = \emptyset$, STOP. Otherwise, consider the column with minimum cost in P and choose at most m customers that are visited more than once. Let C_2 be the set of these customers. If $C_2 = \emptyset$, then add the column to set C_G and return to step 3. Otherwise, set $C_2 \rightarrow C_1 \cup C_2$.
5. Do $P \rightarrow$ 2-Cyc-SPPRC-PE(C_2) until $P = \emptyset$.

Figure 6.1: The main steps of R-COLGEN(m)

Column Generation With the Relaxations of the Pricing Problem. We design a column generation algorithm that uses 2-Cyc-SPPRC and 2-Cyc-SPPRC-PE(C) in combination. The algorithm is referred to as R-COLGEN(m) and described in Figure 6.1.

Variations of the Branch-and-Price Algorithm. For computational testing, we implemented four variants of the branch-and-price algorithm based on the above pricing schemes. The first three designs are also applied for the LRSP case.

- *Heuristic Branch-and-Price* (HBP): The purpose of this algorithm is to provide a good upper bound. At each node of the tree, we use H-COLGEN(u, U, l, L) with small values of u, l and L.
- *One-Stage Exact Branch-and-Price* (1S-EBP): The purpose of this algorithm is to prove the optimality of the solution or provide an integrality gap. At each node of the tree, we use E-COLGEN(u, U, l, L).

- *Two-Stage Exact Branch-and-Price* (2S-EBP): In this variant, HBP is run first to generate initial columns and an upper bound. Then, in the second stage an exact branch-and-price algorithm which is initiated with the columns and the upper bound obtained from HBP is run using E-COLGEN(u, U, l, L).

- *Non-elementary Exact Branch-and-Price (NEBP)*: This is similar to one-stage branch-and-price algorithm except that R-COLGEN(m) is used to solve the pricing problem.

6.3 Computational Results

In this section, we discuss the performance of our branch-and-price algorithms for the LRP on two sets of instances. The first set of instances are random instances that we generated to test the performance of our 1S-EBP and 2S-EBP algorithms. We evaluate the effect of facility capacity constraints and other parameters using this set of instances. The second set contains the LRP instances used in Barreto et al. (2007) that are available from Barreto (2003). We used these instances to test the performance of our HBP and NEBP algorithms. For all of our experiments, we used a Linux-based workstation with a 1.8 GHz processor and 2GB RAM.

6.3.1 Generated Instances

On random instances, Laporte et al. (1986) provided computational results for an exact method (branch-and-cut algorithm) for the capacitated LRP. Belenguer et al. (2006) also developed a branch-and-cut algorithm for the capacitated LRP, but neither the details of the instances nor the computational results are publicly available. Therefore, we evaluated our algorithm by generating random instances as in Laporte et al. (1986), where they generated instances with 10, 15 and 20 customers and 4 to 8 facilities. In addition to these, we generated instances with 30 and 40 customers and 5 facilities to test the performance of the algorithm on larger instances. The coordinates of the customers and the facilities and the demand of each customer were generated using a uniform distribution on [0,100]. We then calculated the Euclidean distance between each

Table 6.1: Details for the instances

# of Customers	# of Facilities	N^F	M^F	# of instances	α
10	4	1	3	3	$\{0, 0.25, 0.5, 0.75, 1\}$
15	6	1	4	3	$\{0, 0.25, 0.5, 0.75, 1\}$
20	8	1	5	3	$\{0, 0.25, 0.5, 0.75, 1\}$

pair of customers and between customers and facilities and rounded the calculated distance to two decimal places. Demand for each customer was rounded to the nearest integer. Vehicle capacity C^V was calculated as

$$C^V = (1-\alpha)\text{max}_{i\in I}\{D_i\} + \alpha \sum_{i\in I} D_i, \tag{6.22}$$

where α was a parameter and the values were chosen in set $\{0, 0.25, 0.5, 0.75, 1\}$.

Small and Medium Random Instances

Laporte et al. (1986) solved location and routing problems with capacitated vehicles, but they did not have a facility capacity. Instead, they had a lower and upper bound for the total number of facilities that could be open in a solution. In this experiment, in order to provide a better comparison of our algorithm with that of Laporte et al. (1986), we removed constraint (6.3) from (SPP-LRP). We set N^F, the minimum number of open facilities in (6.17), to 1, and we added constraint $\sum_{j\in J} t_j \leq M^F$, where M^F is the maximum number of facilities that can be open in any solution. Facility and vehicle fixed costs were set to be zero. As in Laporte et al. (1986), three groups of instances with different numbers of customers and facilities were available. For each group, five different vehicle capacities were calculated by changing α. Some details about the instances are listed in Table 6.1.

Tables 6.2, 6.3 and 6.4 present the results achieved with our branch-and-price algorithms. Instances listed in these tables are labeled with the number of customers and facilities and with letters {a,b,c,d,e} based on the α value used. For example, the instance r10x4-a-1 has 10 customers, 4 facilities and a vehicle capacity calculated with $\alpha = 0$. The integer from 1 to 3 used

as the last character in the label represents the id of the instances within the same group. The tables present the name of each instance, the LP solution value at the root node, the optimal solution value, the number of evaluated nodes, and the CPU time in seconds. In these instances, we first ran the 1S-EBP algorithm for 5 minutes and if the algorithm did not terminate within 5 minutes, we switched to the 2S-EBP algorithm. Table 6.2 presents the results for instances with 10 customers, Table 6.3 presents the results for instances with 15 customers, and Table 6.4 presents the results for instances with 20 customers. We marked the instances with a "+" sign if the 2S-EBP algorithm was used. For instances with at least 20 customers, we needed to use the 2S-EBP algorithm. An exact branch-and-price algorithm (one-stage or two-stage) was very successful in finding an optimal solution quickly. In general, our computation times were much smaller than those reported by Laporte et al. (1986), but it is difficult to make a fair comparison, given advances in computing technology. In most of the instances, the LP solution found by our algorithm at the root node was an optimal integer solution.

The instances become more difficult when the vehicle capacity is increased (α increases) because the number of labels generated in the pricing problem depends directly on the vehicle capacity. Laporte et al. (1986) observed the reverse effect with regard to their branch-and-cut algorithm. The number of cuts generated increases and the problem gets more difficult when the vehicle capacity is small. This is most likely due to the fact that the problem structure becomes more like that of the traveling salesman problem (TSP) as the capacity is increased and the TSP is much easier to solve by branch and cut than as a capacitated routing problem.

To strictly differentiate the instances from those with a single depot, we experimented with changing the value of parameter N^F, the minimum number of open facilities, to 2 and ran the r15x6 and r20x8 c and d instances. There were no significant changes in the computational times or the number of evaluated nodes. We present these results in Table 6.5.

We then added facility capacities to the same set of problems and ran our algorithm again. Table 6.6 presents the computational results, as well as the facility capacity values used for the facilities. The capacity value was chosen in order to require at least two open facilities. The

Table 6.2: Performance of Exact Branch-and-Price for 10-customer instances

Instance	LP	OPT.	# of Nodes	CPU(s)
r10x4-a-1	472.11	472.11	1	0.00
r10x4-a-2	421.44	421.44	1	0.00
r10x4-a-3	548.28	548.28	1	0.02
r10x4-b-1	313.01	313.18	3	0.04
r10x4-b-2	297.57	305.27	19	0.08
r10x4-b-3	352.66	354.92	3	0.03
r10x4-c-1	257.25	257.25	1	0.06
r10x4-c-2	259.76	259.76	1	0.04
r10x4-c-3	296.82	296.82	1	0.05
r10x4-d-1	243.42	257.25	21	0.52
r10x4-d-2	250.04	250.04	1	0.04
r10x4-d-3	296.82	296.82	1	0.04
r10x4-e-1	226.46	226.46	1	0.32
r10x4-e-2	225.82	225.82	1	0.17
r10x4-e-3	272.85	272.85	1	0.31

computational results do not show any significant difference from those of the uncapacitated instances. For larger instances, we expect that the LP solution times will tend to increase if the master problem has facility capacity constraints. Adding a facility capacity to a two-index vehicle flow formulation requires an additional set of constraints (Belenguer et al., 2006), the size of which can be large. In a branch-and-cut algorithm, it may require additional time to generate this set of constraints.

Laporte et al. (1986) report that adding facility fixed costs to the problem makes the problem easier. We added facility fixed costs ($200 for each facility) to r15x6 and r20x8 c and d instances. The performance of the branch-and-price algorithm was not affected in instances with 15 customers. However, for some of the 20-customer instances, the computational times exceeded 1 CPU hour, which is more than the times in the experiments without the fixed cost. Results of this experiment is reported in Table 6.7.

Table 6.3: Performance of Exact Branch-and-Price for 15-customer instances

Instance	LP	OPT.	# of Nodes	CPU(s)
r15x6-a-1	435.2	435.2	1	0.01
r15x6-a-2	663.32	663.32	1	0.01
r15x6-a-3	411.45	411.45	1	0.01
r15x6-b-1	313.46	313.46	1	0.31
r15x6-b-2	414.65	414.65	1	0.21
r15x6-b-3	285.01	285.01	1	0.12
r15x6-c-1	313.46	313.46	1	3.1
r15x6-c-2	392.75	392.75	1	1.98
r15x6-c-3	279.82	279.82	1	5.37
r15x6-d-1	313.36	313.46	3	4.92
r15x6-d-2	378.76	378.76	1	9.41
r15x6-d-3	279.82	279.82	1	14.13
r15x6-e-1	305.86	312.18	5	9.82
r15x6-e-2^{+}	374.86	374.86	1	16.62^{+}
r15x6-e-3	274.22	274.22	1	300.01

$^{+}$ The 2S-EBP algorithm was used, total time is reported.

Large Random Instances

In this section, we present the results of our exact branch-and-price algorithms applied to solve larger capacitated random instances. We generated 6 instances with 30 customers and 6 instances with 40 customers. Each instance had 5 facilities with capacity constraints. The facilities had a fixed cost of 100. The characteristics of each instance are listed in Table 6.8. The first column includes the name of each instance which was set based on the number of customers, facilities and the vehicle capacity. For instances in group "a" the vehicle capacity value (listed in the fourth column) was chosen to be seven times the average demand and for group "b", the vehicle capacity value was 5.5 times the average demand. Facility capacity (listed in the second column) was chosen based on the total demand such that at least two facilities (the minimum number of facilities is listed in the third column) should be open in a feasible solution.

We used the 2S-EBP algorithm in which each stage was run with a time limit of 3 CPU hours. Table 6.9 presents the results of both stages. The algorithm was very successful in finding optimal or near-optimal solutions for these larger instances. Some details, such as the number of open

Table 6.4: Performance of Exact Branch-and-Price for 20-customer instances

Instance	LP	OPT.	# of Nodes	CPU(s)
r20x8-a-1	639.77	653.11	57	0.41
r20x8-a-2	542.23	551.58	25	0.98
r20x8-a-3	760.42	760.42	1	0.05
r20x8-b-1	415.3	417.13	7	8.35
r20x8-b-2	383.19	383.19	1	6.32
r20x8-b-3	439.18	447.72	121	37.32
r20x8-c-1$^+$	398.34	398.34	1	26.25$^+$
r20x8-c-2$^+$	363.86	363.86	1	110.87$^+$
r20x8-c-3$^+$	402.85	402.85	1	28.74$^+$
r20x8-d-1$^+$	392.26	392.26	1	10.04$^+$
r20x8-d-2$^+$	359.49	359.49	1	200.29$^+$
r20x8-d-3$^+$	402.85	402.85	1	124.24$^+$
r20x8-e-1$^+$	392.28	392.28	1	12.39$^+$
r20x8-e-2$^+$	355.39	355.39	1	82.77$^+$
r20x8-e-3$^+$	402.46	402.46	1	103.27$^+$

$^+$ The 2S-EBP algorithm is used, total time is reported.

facilities, the number of vehicles used at each open facility, the average number of customers in each route (vehicle), and the number of customers in the longest route, are presented in Table 6.8.

6.3.2 Instances From the Literature

To the best of our knowledge, there are no benchmark instances available specifically for the LRP. Barreto et al. (2007) used the instances in the literature available for other types of problems to construct a set of LRP instances. They report lower bounds found by applying a branch-and-cut algorithm to the two-index formulation of the problem (Barreto, 2004) and upper bounds found by applying a sequential heuristic based on clustering techniques (Barreto et al., 2007). They listed 19 instances, three of which have more than 150 customers, too large for our approach to work efficiently. We removed these three instances plus one more with 117 customers and fractional demand, since we assume integer demands. The labels of the instances denote the source of the instance and the number of customers and facilities in the instances (for more details about the

Table 6.5: Instances with $N^F \geq 2$

Instance	LP	OPT.	# of Nodes	Total Time
r15x6-c-1	313.46	313.46	1	1.51
r15x6-c-2	392.75	392.75	1	1.28
r15x6-c-3	279.82	279.82	1	4.02
r15x6-d-1	313.36	313.46	3	2.79
r15x6-d-2	378.76	378.76	1	9.45
r15x6-d-3	279.82	279.82	1	19.16
r20x8-c-1	398.34	398.34	1	27.86
r20x8-c-2	363.86	363.86	1	71.9
r20x8-c-3	402.85	402.85	1	19.52
r20x8-d-1	392.26	392.26	1	25.56
r20x8-d-2	359.49	359.49	1	254.83
r20x8-d-3	402.85	402.85	1	124.6

references, see Barreto (2003)).

We first ran HBP with a time limit of 3 CPU hours, focusing on producing quality upper bounds. Table 6.10 presents the instances we tested and compares the results with the upper bounds reported in Barreto et al. (2007). Since neither our HBP nor Barreto et al. (2007) can provide a valid lower bound for the problem, we used the best lower bounds as found in Barreto (2004) (second column in Table 6.10) to measure the quality of our upper bound. The "Gap" in Table 6.10 is the percent gap between the upper bound and the lower bound listed in the second column. HBP is capable of finding better upper bounds (usually optimal) for the instances of small and medium size. In these cases, the computation time is also very short. However, for larger instances (instances with more than 75 customers), the upper bounds reported by Barreto et al. (2007) are generally better. In addition, their heuristic algorithm is very efficient—they report that in most of the instances, it provides the result in less than one second.

Next, we used the 2S-EBP algorithm to test the ability to produce lower bounds and prove optimality. With this algorithm, we could not solve all instances within the total time limit of 5 CPU hours (3 CPU hours for HBP and 2 CPU hours for 1S-EBP). We report the results of five instances that we could solve. The lower bounds found by our algorithm, along with the best integer solution found, optimality gap, and computation time are reported in Table 6.11. Note that

Table 6.6: Instances with Capacitated Facilities

Instance	Fac. Cap	LP	OPT.	# of Nodes	Total Time
cr15x6-c-1	600	299.07	299.07	1	1.15
cr15x6-c-2	600	392.75	392.75	1	2.53
cr15x6-c-3	600	279.82	279.82	1	1.86
cr15x6-d-1	700	299.07	299.07	1	0.57
cr15x6-d-2	700	378.76	378.76	1	6.42
cr15x6-d-3	700	279.82	279.82	1	11.01
cr20x8-c-1	750	398.34	398.34	1	40.46
cr20x8-c-2	750	363.86	363.86	1	88.12
cr20x8-c-3	750	402.85	402.85	1	15.25
cr20x8-d-1	900	392.29	392.29	1	29.1
cr20x8-d-2	750	359.49	359.49	1	257.94
cr20x8-d-3	900	402.85	402.85	1	31.3

because Gaskell67-21x5 and Perl83-12x2 are very easy to solve, we used the 1S-EBP algorithm instead of the 2S-EBP algorithm in these cases.

Finally, for instances that we could not provide lower bounds by using the 1S-EBP or the 2S-EBP algorithm, we used the NEBP algorithm with a time limit of 5 CPU hours or evaluated node limit of 50 nodes. The results are reported in Table 6.12. For the instances C69-100x10, Min92-134x8 and Dsk95-88x8, we could not even solve the root node within the time limit.

In general, the lower bounds found by using branch-and-cut (Barreto, 2004) are better than our lower bounds. However, in some cases, their computation times are much larger than ours. The HBP algorithm can provide good upper bounds, but for instances of medium and large size, we need to improve our methods of computing lower bound, perhaps by adding dynamic cut generation to our algorithm in order to close the optimality gaps.

Table 6.7: Instances with facility fixed cost

Instance	Best IP	%Gap	# of Nodes	Total Time
r15x6-c-1	612.82	0	1	3.33
r15x6-c-2	632.96	0	1	4.53
r15x6-c-3	527.91	0	1	6.9
r15x6-d-1	592.26	0	1	5.47
r15x6-d-2	612.18	0	21	43.67
r15x6-d-3	504.44	0	1	23.88
r20x8-c-1	655.9	0	29	1764.31^{+}
r20x8-c-2	624.8	0	51	2536.09^{+}
r20x8-c-3	666.59	1.42	108	5406.42^{+}
r20x8-d-1	640.14	1.65	84	4700^{+}
r20x8-d-2	605	0.87	25	4253.48^{+}
r20x8-d-3	641.35	0	75	1720.78^{+}

$^{+}$ The 2S-EBP algorithm is used, total time is reported.

Table 6.8: Characteristics of the Instances and the Optimal Solutions

Instance				OPT/BestIP Solution Info			
Name	Fac. Cap.	N^F	L^V	# of Fac.	# of Vec.	Avg. # of Cust/route	# of Cust. longest route
cr30x5a-1	1000	2	350	2	3,2	6	8
cr30x5a-2	1000	2	350	2	3,2	6	7
cr30x5a-3	1000	2	350	2	3,3	5	7
cr30x5b-1	1000	2	275	2	3,2	6	8
cr30x5b-2	1000	2	275	2	4,3	4.29	7
cr30x5b-3	1000	2	275	2	3,4	4.29	6
cr40x5a-1	1750	2	340	2	3,3	6.67	8
cr40x5a-2	1750	2	390	2	3,4	5.71	8
cr40x5a-3	1750	2	370	2	3,3	6.67	7
cr40x5b-1	1750	2	275	2	3,5	5	7
cr40x5b-2	1750	2	275	2	3,5	5	7
cr40x5b-3	1750	2	325	2	5,3	5	7

Table 6.9: Performance of the 2S-EBP for Instances with 30 and 40 Customers, Capacitated Facilities and Facility Fixed Cost

Instance	HBP		Second Stage, Exact Branch and Price					Total
	IP	CPU(s)	LP	IP	Gap	#N.	CPU(s)	CPU (s)
cr30x5a-1	819.53	43.5	810.29	819.5	0	33	993.3	1036.8
cr30x5a-2	823.49	7202.3	790.49	823.49	2.55	500	10806.5	18008.8
cr30x5a-3	702.29	44.2	687.72	702.19	0	51	917.9	962.1
cr30x5b-1	880.02	164.3	865.47	880.01	0	251	6420.6	6585
cr30x5b-2	825.32	8.3	815.95	825.3	0	7	33.2	41.5
cr30x5b-3	884.62	13.4	881.33	884.55	0	19	41.7	55.1
cr40x5a-1	928.11	631.8	911.39	928.11	1.49	11	10882.8	11514.6
cr40x5a-2	888.37	378.4	871.66	888.37	0.93	13	11052.9	11431.3
cr40x5a-3	947.24	173	939.54	947.24	0.18	28	10862	11035
cr40x5b-1	1052.07	257.3	1043.62	1052	0	627	8084.6	8342
cr40x5b-2	981.52	60.1	976.88	981.27	0	47	862.5	922.7
cr40x5b-3	964.32	62.6	959.05	964.23	0	45	963	1025.6

Table 6.10: Performance of Heuristic Branch-and-Price

Instance	LB[1]	Barreto et al. (2007)		Heuristic Branch-and-Price		
		UB[2]	%Gap	UB	Gap	CPU(s)
Gaskell67-21x5	424.9*	435.9	2.59	424.9	0	1.2
Gaskell67-22x5	585.1*	591.5	1.09	585.1	0	41.5
Gaskell67-29x5	512.1*	512.1	0	512.1	0	67.7
Gaskell67-32x5	556.5	571.7	2.73	562.3	1.04	10801.8
Gaskell67-32x5-2	504.3*	511.4	1.41	505.8	0.3	85.6
Gaskell67-36x5	460.4*	470.7	2.24	460.4	0	1077.6
C69-50x5	549.4	582.7	6.06	565.6	2.95	239.4
C69-75x10	744.7	886.3	19.01	852.1	14.42	10802.3
C69-100x10	788.6	889.4	12.78	929.5	17.86	10836.6
Perl83-12x2	204*	204	0	204.0	0	0.2
Perl83-55x15	1074.8	1136.2	5.71	1121.8	4.37	10800.0
Perl83-85x7	1568.1	1656.9	5.66	1668.2	6.38	10813.8
Min92-27x5	3062*	3062	0	3062.0	0	4.2
Min92-134x8	-	6238	-	6421.6	-	10850.9
Dsk95-88x8	356.4	384.9	8	390.6	9.58	10808.9

[1] Reported by Barreto (2004), found using branch-and-cut

[2] Reported by Barreto et al. (2007), found using a heuristic

* Branch-and-cut used in Barreto (2004) proves the optimality

Table 6.11: Performance of Two-Stage Exact Branch-and-Price

Instance	LB	OPT/BestIP	%Gap	CPU(s)	Total CPU(s)
Gaskell67-21x5[1]	424.9	424.9	0	3.0	3.0
Gaskell67-22x5	585.1	585.1	0	2999.9	3041.4
Gaskell67-32x5	544.1	562.3	3.24	8453.1	19254.9
Perl83-12x2[1]	204.0	204.0	0	0.2	0.2
Min92-27x5	3062.0	3062.0	0	833.6	837.8

[1] 1S-EBP algorithm is used

Table 6.12: Performance of Non-Elementary Exact Branch-and-Price

Instance	LB	OPT/BestIP	%Gap	CPU(s)
Gaskell67-29x5	441.2	512.1	13.85	14411.5
Gaskell67-32x5-2	494.4	505.8	2.26	5654.8
Gaskell67-36x5	455.5	460.4	1.05	100.1
C69-50x5	526.2	565.6	6.96	2634.1
C69-75x10	693.5	852.1	18.62	8785.8
Perl83-55x15	852.3	1121.8	24.02	319.5
Perl83-85x7	1272.4	1668.2	23.73	1379.6

Chapter 7

Conclusions and Future Directions

In this research, we focused on solving the integrated facility location, vehicle routing and vehicle scheduling problem (LRSP). In jointly optimizing the decisions of facility location, vehicle routing, and route assignment, the LRSP provides solutions that more accurately capture the practical aspects and costs of designing logistics systems. We considered the interdependency between the facility location decision which is a strategic level decision, and route construction and route assignment decisions which are operational level decisions. This results in a better estimate of the total cost that includes the total vehicle fixed and operating costs, which become crucial in the long term.

We developed mathematical formulations for the LRSP and showed empirically that the set partitioning-based formulation resulted in stronger lower bounds. This is one of the motivations for us to develop the solution algorithm based on the set partitioning-based formulation. We could achieve further reduction in the LP relaxation gap by introducing valid inequalities at the root node. We developed a branch-and-price algorithm with several variants: one-stage exact branch-and-price, two-stage exact branch-and-price and heuristic branch-and-price algorithms.

We computationally evaluated the performance of the branch-and-price algorithm and tried to decide some strategies and parameter values in order to improve the overall performance of the algorithm. As in general, the efficiency of the pricing algorithm significantly affects the overall

performance. We extended our pricing algorithm to a two-phase pricing algorithm and achieved significant improvement (about 90% time reduction at the root node) in the efficiency. In addition, we applied the strategies of adding multiple columns in any iteration of the column generation algorithm and including the heuristic versions of the pricing algorithm and improved the performance of the branch-and-price algorithm further. Furthermore, we presented computational results for instances including 25 and 40 customers. The algorithm found optimal solutions for the majority of the instances and obtained good solutions with small integrality gaps for the rest. Finally, we empirically demonstrated that two-stage exact branch-and-price algorithm was preferred to one-stage exact branch-and-price algorithm while optimizing larger size instances, and the heuristic branch-and-price algorithm was able to provide very strong upper bounds within reasonable times.

We studied the extension of the branch-and-price algorithm to a branch-cut-and-price algorithm using dynamic cut generation. We investigated some classes of valid inequalities that can be used for the LRSP. Even though the cut generation procedures were fast and the valid inequalities we generated improved the LP relaxation bounds at the associated nodes, in our computational experiments we could not observe a performance improvement when we ran branch-cut-and-price algorithms. Furthermore, we studied the disjunctive cut procedure and proposed a methodology that eliminates the major challenges in the procedure that appear in a branch-cut-and-price algorithm. We constructed four implementations of this methodology by choosing different base models to derive valid inequalities and evaluated those computationally. We observed that the efficiency of the disjunctive cut procedure implemented depended on the constructed cut generation linear program formulated based on the base model. We could not observe a performance improvement when we generated disjunctive valid inequalities in the branch-cut-and-price algorithm.

There were two practical implications of our study. First, we computationally demonstrated that there is value in solving the integrated problem. Namely, for a fixed set of customers and candidate facility locations, the integrated problem identifies opportunities to lower cost through

changes in facility selection, reduction in vehicle quantity, and/or changes in customer routing. We could obtain lower costs by solving the LRSP than those obtained by using a sequential optimization of the decisions. Second, since the LRSP is a generalization of other well-known problems, such as the LRP, the MDVRP, the VRP, and the VRPMT, we could modify the algorithm for the LRSP to solve these special cases. To solve the capacitated LRP, we developed a branch-and-price algorithm, which to our knowledge is the first of its kind for this class of problem. We provided computational results for randomly generated instances and instances from the literature. Our experiments indicated that the algorithm is very effective at producing quality solutions and can handle larger instances than previously suggested approaches, which have been primarily based on two-index formulations. The approach, however, does not seem as effective at producing quality lower bounds.

Since we expected that the addition of cutting plane routines would prove helpful in solving larger instances, one area of future research is to explore new inequalities for the LRSP and to explore new ideas to improve the methodology that generates disjunctive cuts in a branch-cut-and-price algorithm. We believe that such a study would be valuable to the literature. Another research avenue is to extend the computational study evaluating the value of integrated optimization in order to generalize the results. As we studied the LRP, we plan to explore possible contributions of our algorithm to the literature for other special cases such as the VRPMT and the MDVRP. In addition to this future work, we plan to investigate possible approaches to improving the performance of our branch-and-price algorithm for the LRSP. One approach is to add another set of branching disjunctions to the set of disjunctions that we applied. We observed that branching on flow between pairs of customer nodes (modified Ryan and Foster branching) was causing inefficiencies in the algorithm. We may search alternative ways to the current method or develop additional branching methods before applying this. Another approach is to develop additional heuristic or exact versions of the pricing algorithm in order to obtain performance improvement. Finally, we have some alternative formulations in mind which would result in better performance of the branch-and-price or branch-cut-and-price algorithms.

Bibliography

K Aardal. Capacitated Facility Location: Separation Algorithms and Computational Experience. *Mathematical Programming*, 81(2):149–175, 1998.

K Aardal, Y. Pochet, and L. A. Wolsey. Capacitated Facility Location: Valid Inequalities and Facets. *Mathematics of Operations Research*, 20:562–582, 1995.

Y. Agarwal, K. Mathur, and H. M. Salkin. A Set-Partitioning-Based Exact Algorithm for the Vehicle Routing Problem. *Networks*, 19:731–749, 1989.

R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra, and M. G. Scutellá. A Multi-exchange Heuristic for the Single-source Capacitated Facility Location Problem. *Management Science*, 50(6):749–760, 2004.

M. Albareda-Sambola, J. A. Diaz, and E. Fernandez. A Compact Model and Tight Bounds for a Combined Location-routing Problem. *Computers and Operations Research*, 32:407–428, 2005.

R. Anbil, J. J. Forrest, and W. R. Pulleyblank. Column Generation and the Airline Crew Pairing Problem. *Proceedings of the International Congress of Mathematicians Berlin*, Extra Volume, III:667–686, 1998.

J. A. Appleget and R. K. Wood. Explicit-Constraint-Branching for Solving Mixed-Integer Programs. In M. Laguna and J.L. González-Velarde, editors, *Computing Tools for Modeling, Optimization and Simulation*, pages 245–261. Kluwer Academic Publishers, Boston, 2000.

J. R. Araque. Lots of Combs of Different Size for Vehicle Routing, 1990. CORE Discussion Paper.

J. R. Araque, L. A. Hall, and T. L. Magnanti. Capacitated Trees, Capacitated Routing and Associated Polyhedra, 1990. CORE Discussion Paper.

P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, and D. Naddef. Separating Capacity Inequalities in the CVRP Using Tabu Search. *European Journal of Operational Research*, 106: 546–557, 1999.

B. M. Baker. A Partial Dual Algorithm for the Capacitated Warehouse Location Problem. *European Journal of Operational Research*, 23:48–56, 1986.

E. Balas. Disjunctive Programming: Facets of the Convex Hull of Feasible Points. Technical report No. 348, GSIA, Carnigie Mellon University, 1974.

E. Balas. Facets of the Knapsack Polytope. *Mathematical Programming*, 8:146–164, 1975.

E. Balas. Disjunctive Programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

E. Balas, S. Ceria, and G. Cornuéjols. A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs. *Mathematical Programming*, 58:295–324, 1993.

E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0-1 Programming by Lift-and-Project in a Branch-and-Cut Framework. *Management Science*, 42(9):1229–1246, 1996.

E. Balas and M. Padberg. Set Partitioning: A Survey. *SIAM Review*, 18:710–760, 1976.

E. Balas and A. Saxena. Optimizing Over the Split Closure. *Mathematical Programming*, 113(2): 219–240, 2008.

M. Balinski and R. Quandt. On an Integer Program for a Delivery Problem. *Operations Research*, 12:300–304, 1964.

C. Barnhart, E. L. Johnson, G.L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch and Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46(3): 316–329, 1998.

S. Barreto. LRP instances, 2003. URL `http://sweet.ua.pt/iscfl43`.

S. Barreto. *Analysis and Modeling of Location-Routing Problems (in Portuguese)*. Phd thesis, University of Aveiro, Aveiro, Portugal, 2004.

S. Barreto, C. Ferreira, J. Paixao, and B. S. Santos. Using Clustering Analysis in a Capacitated Location-routing Problem. *European Journal of Operational Research*, 127(3):968–977, 2007.

J. E. Beasley. An Algorithm for Solving Large Capacitated Warehouse Location Problems. *European Journal of Operational Research*, 33:314–325, 1988.

J. E. Beasley and N. Christofides. An Algorithm for the Resource Constrained Shortest Path Problem. *Networks*, 19:379–394, 1989.

J. M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. Wolfler-Calvo. A Branch and Cut Method for the Capacitated Location-Routing Problem. *Service Systems and Service Management, 2006 International Conference on*, 2:1541–1546, 2006.

R. T. Berger. *Location-Routing Models for Distribution System Design*. PhD thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, 1997.

R. T. Berger, C. R. Coullard, and M. S. Daskin. Location-Routing Problems with Distance Constraints. *Transportation Science*, 41:29–43, 2007.

A. A. Bertossi, P. Carraresi, and G. Gallo. On Some Matching Problems Arising in Vehicle Scheduling Models. *Networks*, 17:271–281, 1987.

L. Bianco, A. Mingozzi, and S. Ricciardelli. A Set Partitioning Approach to the Multiple Depot Vehicle Scheduling Problem. *Optimization Methods and Software*, 3:163–194, 1994.

O. Bilde and J. Krarup. Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem. *Discrete Mathematics*, 1:79–97, 1977.

A. Bixby. *Polyhedral Analysis and Effective Algorithms for the Capacitated Vehicle Routing Problem*. PhD thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, 1999.

A. Bixby, C. Coullard, and D. Simchi-Levi. The Capacitated Prize-collecting Traveling Salesman Problem. Working paper, 1997.

U. Blasum and W. Hochstättler. Application of the Branch and Cut Method to the Vehicle Routing Problem. Technical Report ZPR2000-386, Universität zu Köln, 2000.

L. Bodin and B. Golden. Classification in Vehicle Routing and Scheduling. *Networks*, 11(2): 97–108, 1981.

L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and Scheduling of Vehicles and Crews: The State of the Art. *Computers and Operations Research*, 10:63–211, 1983.

J. Brandao and A. Mercer. A Tabu Search Algorithm for the Multi-Trip Vehicle Routing and Scheduling Problem. *European Journal of Operational Research*, 100:180–191, 1997.

J. Brandao and A. Mercer. The Multi-Trip Vehicle Routing Problem. *Journal of the Operational Research Society*, 49:799–805, 1998.

L. Burlett. Regional Operating Expenses: A Case Study, 2002. Available at http://www.sbd.com/issues/winter2002/features/save.asp.

A. Caprara and A. N. Letchford. On the Separation of Split Cuts and Related Inequalities. *Mathematical Programming*, 94:279–294, 2003.

G. Carpaneto, M. Dell'Amico, M. Fischetti, and P. Toth. A Branch and Bound Algorithm for the Multiple Depot Vehicle Scheduling Problem. *Networks*, 19:531–548, 1989.

P. Carraresi and G. Gallo. Network Models for Vehicle and Crew Scheduling. *European Journal of Operational Research*, 16:139–151, 1984.

I.M. Chao, B. L. Golden, and E. A. Wasil. A New Heuristic for the Multi-Depot Vehicle Routing Problem that Improves Upon Best-Known Solutions. *American Journal of Mathematical and Management Sciences*, 13:371–406, 1993.

D. C. Cho, E. L Johnson, M. W. Padberg, and M. R. Rao. On the Uncapacitated Plant Location Problem. I. Valid Inequalities and Facets. *Mathematics of Operations Research*, 8:579–589, 1983 a.

D. C. Cho, M. W. Padberg, and M. R. Rao. On the Uncapacitated Plant Location Problem. II. Facets and Lifting Theorems. *Mathematics of Operations Research*, 8:590–612, 1983 b.

N. Christofides. Vehicle Routing. In E. L. Lawler, J. F. Lenstra, A.H.G. Rinnooy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem*, pages 431–448. Wiley, Chichester, 1985.

V. Chvátal. Edmonds Polytopes and a Hierarchy of Combinatorial Problems. *Discrete Mathematics*, (4):305–337, 1973.

G. Clarke and J. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581, 1964.

Kenworth Truck Company. White Paper On Life Cycle Cost, 2003.

W. Cook, , R. Kannan, and A. Schrijver. Chva´tal Closures For Mixed Integer Programming Problems. *Mathematical Programming*, 47(2):155–174, 1990.

J. F. Cordeau, M. Gendreau, and G. Laporte. A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. *Networks*, 30:105–119, 1997.

G. Cornuejols. Valid Inequalities For Mixed Integer Linear Programs. *Mathematical Programming B*, 112:2008, 2006.

G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms. *Management Science*, 23:789–810, 1977.

G. Cornuejols and F. Harche. Polyhedral Study of the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 60:21–52, 1993.

G. Cornuejols and Y. Li. Elementary Closures For Integer Programs. *Operations Research Letters*, 28:1–8, 2001.

G. Cornuejols, G. L. Nemhauser, and L.A. Wolsey. The Uncapacitated Facility Location Problems. In P. B. Mirchandani R. L. Francis, editor, *Discrete Location Theory*, pages 119–171. Wiley-Interscience, New York, 1990.

G. Cornuejols and J. M. Thizy. Some Facets of the Simple Plant Location Polytope. *Mathematical Programming*, 23:50–74, 1982.

H. Crowder, E. L. Johnson, and M. Padberg. Solving Large-Scale Zero-One Programming Problems. *Operations Research*, 31(5):803–834, 1983.

F. Cullen, J. Jarvis, and D. Ratliff. Set Partitioning Based Heuristics for Interactive Routing. *Networks*, 11:125–144, 1981.

G. B. Dantzig and P. Wolfe. Decomposition Principle For Linear Programs. *Operations Research*, 8:101–111, 1960.

M. S. Daskin. *Network and Discrete Location: Models, Algorithms and Applications*. John Wiley and Sons, Inc., New York, 1995.

I. R. de Farias and G. L. Nemhauser. A Family of Inequalities for the Generalized Assignment Polytope. *Operations Research Letters*, 29:49–55, 2001.

M. Dell'Amico, M. Fischetti, and P. Toth. Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem. *Management Science*, 39(1):115–125, 1993.

H. Delmaire, J. A. Diaz, E. Fernandez, and M.Ortega. Comparing New Heuristics for the Pure Integer Capacitated Location Problem. Research Report DR97/10, Universidad Politecnica de Catalunya, Spain, 1997.

M. Desrochers. An Algorithm for the Shortest Path Problem with Resource Constraints. Technical Report G-88-27, GERAD, 1988.

M. Desrochers, J. Desrosiers, and M. M. Solomon. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, 40(2):342–354, 1992.

M. Desrochers and G. Laporte. Improvements and Extensions to Miller-Tucker-Zemlin Subtour Elimination Constraints. *Operations Research Letters*, 10:27–36, 1991.

M. Desrochers and F. Soumis. A Column Generation Approach to the Urban Transit Crew Scheduling Problem. *Transportation Science*, 23(1):1–13, 1989.

J. Desrosiers, Y. Dumas, M. Desrochers, F. Soumis, F. Sanso, and P. Trudeau. A Breakthrough in Airline Crew Scheduling. Technical Report G-91-11, GERAD, 1991.

E. D. Dolan and J. J. More. Benchmarking Optimization Software With Performance Profiles. *Mathematical Programming*, 91(2):201–213, 2002.

M. Dror. Note on the Complexity of the Shortest Path Models for Column Generation in vrptw. *Operations Research*, 42(5):977–978, 1994.

I. Dumitrescu. *Constrained Path and Cycle Problems*. PhD thesis, Department of Mathematics and Statistics, University of Melbourne, 2002.

S. Eilon. Management Perspectives in Physical Distribution. *Omega*, 5(4):437–462, 1977.

D. Erlenkotter. A Dual Based Procedure for Uncapacitated Facility Location. *Operations Research*, 26:992–1009, 1978.

M. Eso. *Parallel Branch-and-cut for Set Partitioning*. Phd thesis, Cornell University, Ithaca, NY, 1999.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to Some Vehicle Routing Problems. *Networks*, 44(3):216–229, 2004.

M. Fischetti, A. Lodi, S. Martello, and P. Toth. A Polyhedral Approach to Simplified Crew Scheduling and Vehicle Scheduling Problems. *Management Science*, 47(6):833–850, 2001.

M. A. Forbes, J. N. Holt, and A. M. Watts. An Exact Algorithm for Multiple Depot Bus Scheduling. *European Journal of Operational Research*, 72:115–124, 1994.

R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 106(3):491–511, 2006.

R. Galvao. The Use of Lagrangean Relaxation in the Solution of Uncapacitated Facility Location Problems. *Location Science*, 1:57–79, 1993.

W. M. Garvin, H. W. Crandall, J. B. John, and R. A. Spellman. Applications of Linear Programming in the Oil Industry. *Management Science*, 3:407–430, 1957.

B. Gavish and S. C. Graves. The Traveling Salesman Problem and Related Problems. Working Paper 7905, 1979.

M. Gendreau, A. Hertz, and G. Laporte. New Insertion and Post Optimization Procedures for the Traveling Salesman Problem. *Operations Research*, 40:1086–1094, 1992.

B.E. Gillett and J.G. Johnson. Multi-Terminal Vehicle-Dispatch Algorithm. *Omega*, 4:711–718, 1976.

R. E. Gomory. An Algorithm For Integer Solutions to Linear Programming. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302, New York, 1963. McGraw-Hill.

E. S. Gottlieb and M. R. Rao. The Generalized Assignment Problem: Valid Inequalities and Facets. *Mathematical Programming*, 46:31–52, 1990.

L. Gouveia. A Result On Projection for the Vehicle Routing Problem. *European Journal of Operational Research*, 85(3):610–624, 1995.

Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. Lifted cover inequalities for 0–1 integer programs: computation. *INFORMS Journal on Computing*, 10:427–438, 1998.

M. Guignard. Fractional Vertices, Cuts and Facets of the Simple Plant Location Problem. *Mathematical Programming Study*, 12:150–162, 1980.

E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A New Exact Algorithm for the Vehicle Routing Problem Based on q-Paths and k-Shortest Paths Relaxations. *Annals of Operations Research*, 61:21–43, 1995.

S. L. Hakimi. Optimal Locations of Switching Centers and the Absolute Centers and Medians of a Graph. *Operations Research*, 12:450–459, 1964.

S. L. Hakimi. Optimal Distribution of Switching Centers in a Communication Network and Some Related Theoretic Graph Problems. *Operations Research*, 13:462–475, 1965.

P. H. Hansen, B. Hegedahl, S. Hjortkjaer, and B. Obel. A Heuristic Solution to the Warehouse Location-Routing Problem. *European Journal of Operational Research*, 76:111–127, 1994.

K. Holmberg, M. Rönnqvist, and D. Yuan. An Exact Algorithm for the Capacitated Facility Location Problem with Single Sourcing. *European Journal of Operational Research*, 113:544–559, 1999.

S. Irnich and G. Desaulniers. *Column Generation*, chapter Shortest Path Problems with Resource Constraints, pages 33–65. GERAD 25th Anniversary Series. Springer, 2005.

M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows. *Operations Research*, 56:497–511, 2008.

O. Kariv and S.L. Hakimi. An Algorithmic Approach to Network Location Problems. Part ii: The p-median. *SIAM Journal of Applied Mathematics*, 37:539–560, 1979.

J. Klincewicz and H. Luss. A Lagrangean Relaxation for Capacitated Facility Location with Single-source Constraints. *Journal of the Operational Research Society*, 37:495–500, 1986.

A. Klose and A. Drexl. Facility Location Models for Distribution System Design. *European Journal of Operational Research*, 162:4–29, 2005.

M. Körkel. On the Exact Solution of Large Scale Simple Plant Location Problems. *European Journal of Operational Research*, 39:157–173, 1989.

M. Labbé and F. V. Louveaux. Location problems. In S. Martello M. Dell'Amico, F. Maffioli, editor, *Annotated Bibliographies in Combinatorial Optimization*. J. Wiley & Sons, New York, 1997.

G. Laporte. Location Routing Problems. In B.L. Golden and A.A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 293–318. North-Holland, Amsterdam, 1988.

G. Laporte. The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research*, 59:345–358, 1992.

G. Laporte and Y. Nobert. An Exact Algorithm for Minimizing Routing and Operating Cost in Depot Location. *European Journal of Operational Research*, 6:224–226, 1981a.

G. Laporte and Y. Nobert. Comb Inequalities for the Vehicle Routing Problem. *Methods of Operations Research*, 51:271–, 1981b.

G. Laporte and Y. Nobert. Exact Algorithms for the Vehicle Routing Problem. *Annals of Discrete Mathematics*, 31:147–184, 1987.

G. Laporte, Y. Nobert, and D. Arpin. Optimal Solutions to Capacitated Multidepot Vehicle Routing Problems. *Congressus Numerantium*, 44:283–292, 1984.

G. Laporte, Y. Nobert, and D. Arpin. An Exact Algorithm for Solving a Capacitated Location-Routing Problem. *Annals of Operations Research*, 6:293–310, 1986.

G. Laporte, Y. Nobert, and M. Desrouchers. Optimal Routing with Capacity and Distance Restrictions. *Operations Research*, 33:1050–1073, 1985.

G. Laporte, Y. Nobert, and Y. Pelletier. Hamiltonian Location Problems. *European Journal of Operational Research*, 12:82–89, 1983.

G. Laporte, Y. Nobert, and S. Taillefer. Solving a Family of Multi-Depot Vehicle Routing and Location-Routing Problems. *Transportation Science*, 22:161–172, 1988.

J. Lenstra and A. Rinnooy Kan. Complexity of Vehicle Routing and Scheduling Problems. *Networks*, 11:221–228, 1981.

A. N. Letchford. On Disjunctive Cuts For Combinatorial Optimization. *Journal of Combinatorial Optimization*, 5:299–315, 2001.

A. N. Letchford and J. José S. González. Projection Results for Vehicle Routing. *Mathematical Programming*, 105(2-3):251–274, 2006.

J. M. Y. Leung and T. L. Magnanti. Valid Inequalities and Facets of the Capacitated Plant Location Problems. *Mathematical Programming*, 44:271–291, 1989.

C. K. Lin, C. K. Chew, and A. Chen. A Location-Routing-Loading Problem for Bill Delivery Services. *Computers and Industrial Engineering*, 43:5–25, 2002.

C. K. Lin and R. C. W. Kwok. Multi-Objective Metaheuristics for a Location-Routing Problem with Multiple Use of Vehicles on Real Data and Simulated Data. *European Journal of Operational Research*, 175(3):1833–1849, 2006.

J. Linderoth. Lecture Notes for Integer Programming, Department of Industrial and Systems Engineering, Lehigh University. 2005.

M. E. Lübbecke and J. Desrosiers. Selected Topics in Column Generation. *Operations Research*, 53(6):1007–1023, 2005.

F. Maranzana. On the Location of Supply Points to Minimize Transport Cost. *Operations Research Quarterly*, 15:261–270, 1964.

T. McCormick, M. R. Rao, and G. Rinaldi. When is Min Cut with Negative Edges Easy to Solve? Easy and Difficult Objective Functions for Max Cut. Research report, IASI-CNR, 2002.

C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer Programming Formulations and Traveling Salesman Problems. *Journal of ACM*, 7(1):326–329, 1960.

H. Min. Consolidation Terminal Location-allocation and Consolidated Routing Problems. *Journal of Business Logistics*, 17(2):235–263, 1996.

H. Min, V. Jayaraman, and R. Srivastava. Combined Location-Routing Problems: A Synthesis and Future Research Directions. *European Journal of Operational Research*, 108(1):1–15, 1998.

G. Nagy and S. Salhi. Location-routing Issues, Models and Methods. *European Journal of Operational Research*, 177:649–672, 2007.

J. M. Nambiar, L. F. Gelders, and L. N. Van Wassenhove. A Large Scale Location Allocation Problem in the Natural Rubber Industry. *European Journal of Operational Research*, 6:183–189, 1981.

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, New York, 1989.

G. L. Nemhauser and L. A. Wolsey. A Recursive Procedure to Generate All Cuts For 0-1 Mixed Integer Programs. *Mathematical Programming*, 46(3):379–390, 1990.

A. Olivera and O. Viera. Adaptive Memory Programming for the Vehicle Routing Problem with Multiple Trips. *Computers and Operations Research*, 34:28–47, 2007.

M. W. Padberg. (1,k)- Configurations and Facets for Packing Problems. *Mathematical Programming*, 18:94–99, 1980.

M. W. Padberg, T. J. van Roy, and L. A. Wolsey. Valid Inequalities for Fixed Charge Problems. *Operations Research*, 33:842–861, 1985.

J. Perl and M. S. Daskin. A Warehouse Location-Routing Model. *Transportation Research*, 19 (B):381396, 1985.

M. Perregaard and E. Balas. Generating Cuts from Multiple-term Disjunctions. In *Integer Programming and Combinatorial Optimization*, pages 348–360. Springer Berlin / Heidelberg, 2001.

R. J. Petch and S. Salhi. A Multi-Phase Constructive Heuristic for the Vehicle Routing Problem with Multiple Trips. *Discrete Applied Mathematics*, 133:69–92, 2004.

H. Pirkul. Efficient Algorithms for the Capacitated Concentrator Location Problem. *Computers and Operations Research*, 14:197–208, 1987.

C. Prins, C. Prodhon, A. Ruiz, P. Soriano, and R. Wolfler Calvo. Solving the Capacitated Location-Routing Problem by a Cooperative Lagrangean Relaxation-Granular Tabu Search Heuristic. *Transportation Science*, 41(4):470–483, 2007.

T. K. R. Ralphs, L. Kopman, W. R. Pulleyblank, and L.E. Trotter. On the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 94:343–, 2003.

J. Renaud, G. Laporte, and F. F. Boctor. A Tabu Search Heuristic for the Multi-Depot Vehicle Routing Problem. *Computers and Operations Research*, 23(3):229–235, 1996.

MacroSy Research and Technology. Logistics Costs and U.S. Gross Domestic Product, 2005. Prepared for Federal Highway Administration Department of Transportation. Available at http://ops.fhwa.dot.gov/freight/freight_analysis/econ_methods/lcdp_rep/index.htm.

C. S. ReVelle and H. A. Eiselt. Location Analysis: A Synthesis and Survey. *European Journal of Operational Research*, 165:1–19, 2005.

C. S. ReVelle and R. W. Swain. Central Facilities Location. *Geographical Analysis*, 2:30–42, 1970.

C. Ribeiro and F. Soumis. A Column Generation Approach to the Multiple Depot Vehicle Scheduling Problem. *Operations Research*, 42:41–52, 1994.

Y. Rochat and E. Taillard. Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Heuristics*, 1:147–167, 1995.

S. Ropke. Disjunctive Cuts in Branch-and-Cut-and-Price Algorithms. Application to the Capacitated Vehicle Routing Problem. In *International Workshop on Column Generation*, June 2008.

S. Ropke and J. F. Cordeau. Branch and Cut and Price for Pickup and Delivery Problems With Time Windows. *Transportation Science*, 43(3):267–286, 2009.

K. Rosing and C. S. ReVelle. Heuristic Concentration: Two Stage Solution Construction. *European Journal of Operational Research*, 97:75–86, 1997.

T. J. Van Roy. A Cross Decomposition Algorithm for Capacitated Facility Location. *Operations Research*, 34:145–163, 1986.

D. Ryan and B. Foster. An Integer Programming Approach to Scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pages 269–280. North-Holland, 1981.

S. Salhi and G. K. Rand. The Effect of Ignoring Routes When Locating Depots. *European Journal of Operational Research*, 39(2):150–156, 1989.

M.W.P. Savelsbergh and G. Nemhauser. *Functional Description of MINTO, A Mixed INTeger Optimizer*. Georgia Institute of Technology, 1996.

H. P. Simao and J. M. Thizy. A Dual Simplex Algorithm for the Canonical Representation of the Uncapaciated Facility Location Problem. *Operations Research Letters*, 8:279–286, 1989.

S. Spoorendonk and G. Desaulniers. Clique Inequalities Applied to the Vehicle Routing Problem With Time Windows. In *International Workshop on Column Generation*, June 2008a.

S. Spoorendonk and G. Desaulniers. Clique inequalities applied to the vehicle routing problem with time windows. Les Cahiers des GERAD G-2008-72, GERAD, Canada, 2008b.

R. Sridharan. A Lagrangean Heuristic for the Capacitated Plant Location Problem with Single Source Constraints. *European Journal of Operational Research*, 66:305–312, 1993.

R. Srivastava. Alternate Solution Procedures for the Location-routing Problem. *Omega*, 21(4): 497–506, 1993.

E. D. Taillard, G. Laporte, and M. Gendreau. Vehicle Routing with Multiple Use of Vehicles. *The Journal of Operational Research Society*, 47(8):1065–1070, 1996.

M. Teitz and P. Bart. Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph. *Operations Research*, 16:955–961, 1968.

R. R. Thomas. Lecture Notes for Linear and Integer Polyhedra, Department of Mathematics, University of Washington. 2005.

F. A. Tillman. The Multiple Terminal Delivery Problem with Probabilistic Demands. *Transportation Science*, 3:192–204, 1969.

F. A. Tillman and T. M. Cain. An Upper Bound Algorithm for the Single and Multiple Terminal Delivery Problem. *Management Science*, 18:664–682, 1972.

P. Toth and D. Vigo. *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications, 2002.

P. Toth and D. Vigo. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS J. on Computing*, 15(4):333–346, 2003.

H. P. Vance, C. Barnhart, E. Johnson, and G. L. Nemhauser. Airline Crew Scheduling: A New Formulation and Decomposition Algorithm. *Operations Research*, 45(2):188–199, 1997.

F. Vanderbeck. *Decomposition and Column Generation for Integer Programs*. Phd thesis, Universite Catholique de Louvain, 1994.

M. H. Webb. Cost Functions in the Location of Depots for Multiple-Delivery Journeys. *Operational Research Quarterly*, 19:311–320, 1968.

R. Weismantel. On the $0\backslash 1$ Knapsack Polytope. *Mathematical Programming*, 77:49–68, 1997.

Appendix A

Tables of Computational Results

In tables:

- “-” represents the missing data caused by termination with time limit,
- “x” represents the missing data caused by termination prematurely since memory,
- “*” represents the missing data caused by the termination signal of condor since time,
- “**” refers to the case when the root node cannot be solved within time limit.

Table A.1: LP relaxation bounds with simple valid inequalities: 20-cust. Inst.

Data	# of Pairings	IP Obj	SPP % Gap	SPP_1 % Gap	SPP_2 % Gap	SPP_3 % Gap	ASPP % Gap
p01-f20-v1t1	67558	4420	24.54	24.49	19.79	2.48	1.78
p01-f20-v1t2	154903	4419	26.06	-	22.08	3.69	3.52
p01-f20-v2t1	147368	4378	26.17	-	23.24	3.80	3.44
p01-f20-v2t2	349885	4369	27.52	-	25.05	4.86	4.83
p01-l20-v1t1	26266	4780	33.51	33.38	27.98	5.37	2.16
p01-l20-v1t2	60211	4599	32.77	32.72	28.08	3.49	0.58
p01-l20-v2t1	51538	4605	33.64	33.51	29.91	4.30	1.57
p01-l20-v2t2	121027	4481	33.24	33.23	30.81	2.97	1.54
p03-f20-v1t1	52619	4468	15.64	15.63	14.08	0.72	0.43
p03-f20-v1t2	121933	4461	17.25	-	16.12	2.32	2.13
p03-f20-v2t1	107172	4421	18.04	-	17.42	2.88	2.75
p03-f20-v2t2	254863	4421	19.78	-	19.12	4.58	4.44
p03-m20-v1t1	37751	4543	18.6	18.42	15.83	3.20	3.02
p03-m20-v1t2	89465	4442	18.13	-	16.02	2.35	2.35
p03-m20-v2t1	78580	4383	19.15	-	17.59	3.19	3.19
p03-m20-v2t2	193509	4376	20.13	-	19.2	4.12	4.12
p03-l20-v1t1	32801	4617	31.27	30.84	26	4.74	1.72
p03-l20-v1t2	79215	4467	30.19	30.13	27.08	2.77	1.30
p03-l20-v2t1	51052	4404	29.84	29.84	28.35	1.45	1.04
p03-l20-v2t2	133853	4399	31.76	-	30.45	3.55	3.11
p07-f20-v1t1	68267	4426	38.22	37.96	34.78	2.81	1.13
p07-f20-v1t2	166551	4421	39.33	-	37.18	3.60	2.57
p07-f20-v2t1	133873	4380	39.45	-	38.55	3.25	2.58
p07-s20-v1t1	27623	4840	31.69	31.47	25.75	4.73	1.01
p07-s20-v1t2	61753	4801	32.79	32.72	27.97	5.61	2.91
p07-s20-v2t1	54964	4751	32.8	32.48	29.09	5.34	2.32
p07-s20-v2t2	127679	4543	31.15	-	28.1	2.43	0.68
p07-t20-v1t1	11267	4758	29.49	29.47	26.03	3.62	1.05
p07-t20-v1t2	28315	4670	30.44	30.06	28.58	3.77	2.08
p07-t20-v2t1	19283	4688	32.39	31.83	30.98	5.57	4.14
p07-t20-v2t2	53689	4460	30.47	30.28	29.2	2.11	1.33
Avg.			28.24	29.91	25.50	3.54	2.28

Table A.2: LP relaxation bounds with simple valid inequalities: 25-cust. Inst.

Data	SPP-LRSP % Gap	SPP-LRSP$_2$ % Gap	SPP-LRSP$_3$ % Gap	ASPP-LRSP % Gap
p01-f25-v1t1	18.66	13.59	5.08	4.32
p01-f25-v1t2	15.91	11.89	2.4	1.93
p01-f25-v2t1	15.33	12.41	1.65	1.35
p01-f25-v2t2	17.6	14.98	3.09	3.02
p01-l25-v1t1	30.66	21.63	4.25	1.01
p01-l25-v1t2	31.88	24.04	4.65	1.9
p01-l25-v2t1	32.42	26.29	4.7	1.93
p01-l25-v2t2	30.62	26.58	2.89	1.43
p03-f25-v1t1	9.26	6.41	7.9	5.24
p03-f25-v1t2	6.16	4.09	4.82	2.89
p03-f25-v2t1	6.55	4.53	5.14	3.3
p03-f25-v2t2	8.32	6.6	6.83	5.32
p03-m25-v1t1	6.77	4.12	3.91	1.87
p03-m25-v1t2	8.78	6.98	5.75	4.52
p03-m25-v2t1	8.18	7.25	5.08	4.53
p03-m25-v2t2	5.48	4.54	2.44	1.89
p03-l25-v1t1	22.54	16.39	3.98	0.97
p03-l25-v1t2	25.11	19.31	5.55	2.84
p03-l25-v2t1	23.88	20.31	4.35	2.6
p03-l25-v2t2	24.74	21.75	4.48	3.27
p07-f25-v1t1	39.27	29.8	5.22	2.55
p07-f25-v1t2	39.43	32.18	4.79	3.09
p07-f25-v2t1	41.78	37.41	6.15	4.85
p07-f25-v2t2	39.42	34.58	3.34	2.13
p07-s25-v1t1	23.23	17.58	5.03	1.96
p07-s25-v1t2	21.72	17.25	3.28	0.76
p07-s25-v2t1	20.69	18.59	2.21	1
p07-s25-v2t2	25.24	22.55	5.42	3.71
p07-t25-v1t1	30.45	21.16	4.84	1.27
p07-t25-v1t2	32.68	25.56	6.22	3.89
p07-t25-v2t1	31.27	26.33	4.8	2.6
p07-t25-v2t2	29.28	25.39	2.74	1.75
Average	22.6	18.19	4.47	2.68

Table A.3: LP relaxation bounds with simple valid inequalities: 40-cust. Inst.

Data	SPP-LRSP % Gap	SPP-LRSP$_2$ % Gap	SPP-LRSP$_3$ % Gap	ASPP-LRSP % Gap
p01-f40-v1t1	18.46	14.6	4.45	2.22
p01-f40-v1t2	17.54	14.86	3.46	2.3
p01-f40-v2t1	17.34	15.84	3.06	2.47
p01-f40-v2t2	18.56	17.67	3.74	3.6
p01-l40-v1t1	19.95	16.73	5.01	3.76
p01-l40-v1t2	18.13	16.01	3.01	2.61
p01-l40-v2t1	18.75	17.74	3.23	3.12
p01-l40-v2t2	17.54	16.64	1.51	1.51
p03-f40-v1t1	21.01	19	1.16	0.95
p03-f40-v1t2	22.64	20.92	2.06	2.03
p03-f40-v2t1	20.24	19.28	0.01	0.01
p03-f40-v2t2	21.5	21.08	0.55	0.55
p03-m40-v1t1	7.18	4.4	5.44	2.87
p03-m40-v1t2	5.72	4.01	4.02	2.42
p03-m40-v2t1	5.88	4.81	4.04	3.08
p03-m40-v2t2	6.78	6.18	4.85	4.38
p03-l40-v1t1	15.43	11.92	5.62	3.56
p03-l40-v1t2	13.43	11.16	3.77	2.64
p03-l40-v2t1	13.51	12.03	3.68	2.79
p03-l40-v2t2	11.72	11.03	1.55	1.44
p07-f40-v1t1	25.5	22.13	3.13	1.62
p07-f40-v1t2	26.01	22.77	2.93	1.37
p07-f40-v2t1	26.65	24.54	2.99	2.01
p07-f40-v2t2	26.03	24.78	2.06	1.2
p07-s40-v1t1	20.11	17.17	2.78	1.19
p07-s40-v1t2	19.62	17.03	1.98	0.37
p07-s40-v2t1	20.66	18.94	2.05	1.09
p07-s40-v2t2	22.65	21.53	3.4	2.68
p07-t40-v1t1	27.9	24.64	2.53	1.03
p07-t40-v1t2	29.85	27.27	3.49	2.43
p07-t40-v2t1	27.58	25.99	0.88	0.23
p07-t40-v2t2	30.07	28.83	2.35	1.88
Average	19.19	17.24	2.96	2.04

Table A.4: Performance of 1p-ESPPRC, 1p-ESPPRC-NwD, 2p-ESSPRC: 25-customer instances

	1p-ESPPRC		1p-ESPPRC-NwD			2p-ESPPRC		
Data File	Tot.time (s) (a)	Time per iter	Tot.time (s) (b)	(b/a)	Time per iter	Tot.time (s) (c)	(c/a)	Time per iter
p01-f25-v1t1	282.44	4.15	174.46	0.62	2.57	21.42	0.08	0.32
p01-f25-v1t2	2555.16	33.18	1238.2	0.48	16.08	91.44	0.04	0.82
p01-f25-v2t1	2001.41	21.75	849.4	0.42	11.48	116.77	0.06	1.27
p01-f25-v2t2	13467.95	134.68	6515.66	0.48	69.32	515.59	0.04	5.49
p01-l25-v1t1	134.5	2.36	76.97	0.57	1.48	14.37	0.11	0.29
p01-l25-v1t2	752.68	12.98	425.1	0.56	6.54	44.32	0.06	0.76
p01-l25-v2t1	451.86	9.41	268.62	0.59	5.6	68.69	0.15	0.85
p01-l25-v2t2	2127.41	31.75	1601.02	0.75	24.63	195.69	0.09	2.8
p03-f25-v1t1	154.57	2.53	77.29	0.5	1.55	15.19	0.1	0.24
p03-f25-v1t2	726.3	10.38	460.92	0.63	6.23	60.06	0.08	0.7
p03-f25-v2t1	616.07	8.21	290.1	0.47	4.4	45.19	0.07	0.65
p03-f25-v2t2	3273.45	32.73	2054.31	0.63	23.08	142.52	0.04	1.76
p03-m25-v1t1	362.8	6.6	178.94	0.49	3.14	24.59	0.07	0.38
p03-m25-v1t2	3238.9	46.27	1427.19	0.44	19.55	107.76	0.03	1.31
p03-m25-v2t1	1714.34	24.49	1117.63	0.65	16.68	66.95	0.04	1.01
p03-m25-v2t2	12680.8	162.57	6380.45	0.5	76.87	321.98	0.03	4.13
p03-l25-v1t1	84.76	1.3	58.64	0.69	1.01	15.32	0.18	0.25
p03-l25-v1t2	499.57	7.35	285.85	0.57	4.2	37.17	0.07	0.51
p03-l25-v2t1	187.57	3.03	145.72	0.78	2.21	41.57	0.22	0.7
p03-l25-v2t2	1417.49	19.96	962.73	0.68	13.56	114.12	0.08	1.52
p07-f25-v1t1	150.75	2.79	155.74	1.03	2.88	26.22	0.17	0.47
p07-f25-v1t2	1154.08	16.73	760.58	0.66	11.02	86.32	0.07	1.2
p07-f25-v2t1	421.47	6.8	327.5	0.78	5.2	98.88	0.23	1.73
p07-f25-v2t2	2231.63	31.88	1664.52	0.75	24.48	293.53	0.13	4.59
p07-s25-v1t1	29.38	0.61	22.13	0.75	0.37	6.28	0.21	0.14
p07-s25-v1t2	119.4	2.3	77.21	0.65	1.54	13.71	0.11	0.3
p07-s25-v2t1	51.11	1.14	44.11	0.86	0.98	19.45	0.38	0.41
p07-s25-v2t2	305.07	4.49	220.66	0.72	3.56	51.44	0.17	0.87
p07-t25-v1t1	162.02	2.7	126.93	0.78	1.98	22.49	0.14	0.39
p07-t25-v1t2	1505.23	21.2	976.62	0.65	14.15	50.21	0.03	0.85
p07-t25-v2t1	641.48	12.1	448.56	0.7	8.16	75.14	0.12	1.45
p07-t25-v2t2	4036.72	61.16	2492.33	0.62	39.56	223.61	0.06	3.29
Average	1798.07	23.11	997.07	0.64	13.25	94.62	0.11	1.3

Table A.5: Performance of 1p-ESPPRC, 1p-ESPPRC-NwD, 2p-ESSPRC: 40-customer instances

	1p-ESPPRC		1p-ESPPRC-NwD			2p-ESPPRC		
Data File	Tot.time (s) (a)	Time per iter	Tot.time (s) (b)	(b/a)	Time per iter	Tot.time (s) (c)	(c/a)	Time per iter
p01-f40-v1t1	13272.35	189.61	9252.84	0.7	118.63	421.91	0.03	5.15
p01-f40-v1t2	-	-	-	-	-	2071.86	-	23.02
p01-f40-v2t1	-	-	-	-	-	2403	-	24.77
p01-f40-v2t2	-	-	-	-	-	-	-	-
p01-l40-v1t1	7822.41	87.89	4749.8	0.61	53.37	305.09	0.04	3.63
p01-l40-v1t2	-	-	-	-	-	1633.48	-	15.12
p01-l40-v2t1	-	-	-	-	-	2627.27	-	24.33
p01-l40-v2t2	-	-	-	-	-	-	-	-
p03-f40-v1t1	-	-	-	-	-	967.01	-	12.56
p03-f40-v1t2	-	-	-	-	-	7304.94	-	73.05
p03-f40-v2t1	-	-	-	-	-	3665.82	-	29.56
p03-f40-v2t2	-	-	-	-	-	-	-	-
p03-m40-v1t1	10858.44	116.76	6065.59	0.56	60.06	372.55	0.03	4.19
p03-m40-v1t2	-	-	-	-	-	2102.19	-	17.97
p03-m40-v2t1	-	-	-	-	-	2080.48	-	20.6
p03-m40-v2t2	-	-	-	-	-	8504.13	-	75.93
p03-l40-v1t1	-	-	-	-	-	728.35	-	6.39
p03-l40-v1t2	-	-	-	-	-	2451.58	-	20.43
p03-l40-v2t1	-	-	-	-	-	1898.16	-	20.63
p03-l40-v2t2	-	-	-	-	-	-	-	-
p07-f40-v1t1	6334.35	86.77	4278.6	0.68	56.3	360.19	0.06	5.22
p07-f40-v1t2	-	-	-	-	-	1673.53	-	22.31
p07-f40-v2t1	-	-	-	-	-	2131.64	-	27.68
p07-f40-v2t2	-	-	-	-	-	11560.98	-	129.9
p07-s40-v1t1	1464.53	23.62	978.07	0.67	16.58	152.88	0.1	2.25
p07-s40-v1t2	9537.53	136.25	6579.21	0.69	96.75	580.9	0.06	7.64
p07-s40-v2t1	3811.24	61.47	2901.68	0.76	46.8	574.03	0.15	8.57
p07-s40-v2t2	-	-	-	-	-	3320.35	-	46.77
p07-t40-v1t1	-	-	-	-	-	1026.73	-	13.33
p07-t40-v1t2	-	-	-	-	-	10006.02	-	125.08
p07-t40-v2t1	-	-	-	-	-	5200.24	-	68.42
p07-t40-v2t2	-	-	-	-	-	-	-	-
Average	7585.84	100.34	4972.26	0.67	64.07	2819.46	0.07	30.91

Table A.6: Performance of 2p-ESSPRC and 2p-ESSPRC$_{40}$: 25-customer instances

	2p-ESPPRC			2p-ESPPRC$_{40}$			
Data File	Tot.time (s) (c)	# of iter	Time per iter	Tot.time (s) (d)	(d/c)	# of iter	Time per iter
p01-f25-v1t1	21.42	68	0.32	12.87	0.6	38	0.34
p01-f25-v1t2	91.44	111	0.82	33.98	0.37	34	1
p01-f25-v2t1	116.77	92	1.27	44.24	0.38	48	0.92
p01-f25-v2t2	515.59	94	5.49	143.92	0.28	52	2.77
p01-l25-v1t1	14.37	49	0.29	7.36	0.51	22	0.33
p01-l25-v1t2	44.32	58	0.76	31.03	0.7	35	0.89
p01-l25-v2t1	68.69	81	0.85	29.54	0.43	27	1.09
p01-l25-v2t2	195.69	70	2.8	95.13	0.49	29	3.28
p03-f25-v1t1	15.19	63	0.24	10.3	0.68	41	0.25
p03-f25-v1t2	60.06	86	0.7	27.4	0.46	35	0.78
p03-f25-v2t1	45.19	69	0.65	27.85	0.62	33	0.84
p03-f25-v2t2	142.52	81	1.76	70.43	0.49	35	2.01
p03-m25-v1t1	24.59	64	0.38	12.88	0.52	25	0.52
p03-m25-v1t2	107.76	82	1.31	45.75	0.42	36	1.27
p03-m25-v2t1	66.95	66	1.01	44.76	0.67	29	1.54
p03-m25-v2t2	321.98	78	4.13	173.35	0.54	38	4.56
p03-l25-v1t1	15.32	62	0.25	7.67	0.5	30	0.26
p03-l25-v1t2	37.17	73	0.51	14.91	0.4	27	0.55
p03-l25-v2t1	41.57	59	0.7	31.49	0.76	31	1.02
p03-l25-v2t2	114.12	75	1.52	76.26	0.67	33	2.31
p07-f25-v1t1	26.22	56	0.47	13.41	0.51	26	0.52
p07-f25-v1t2	86.32	72	1.2	49.83	0.58	34	1.47
p07-f25-v2t1	98.88	57	1.73	45.55	0.46	21	2.17
p07-f25-v2t2	293.53	64	4.59	140.89	0.48	27	5.22
p07-s25-v1t1	6.28	45	0.14	3.26	0.52	18	0.18
p07-s25-v1t2	13.71	45	0.3	7.82	0.57	24	0.33
p07-s25-v2t1	19.45	47	0.41	7.29	0.37	18	0.41
p07-s25-v2t2	51.44	59	0.87	27.7	0.54	31	0.89
p07-t25-v1t1	22.49	57	0.39	12.01	0.53	26	0.46
p07-t25-v1t2	50.21	59	0.85	34.9	0.7	38	0.92
p07-t25-v2t1	75.14	52	1.45	38.68	0.51	22	1.76
p07-t25-v2t2	223.61	68	3.29	124.55	0.56	29	4.29
Average	94.62	67.56	1.3	45.22	0.53	31	1.41

Table A.7: Performance of 2p-ESSPRC and 2p-ESSPRC$_{40}$: 40-customer instances

	2p-ESPPRC			2p-ESSPRC$_{40}$			
Data File	Tot.time (s) (c)	# of iter	Time per iter	Tot.time (s) (d)	(d/c)	# of iter	Time per iter
p01-f40-v1t1	421.91	82	5.15	237.42	0.56	37	6.42
p01-f40-v1t2	2071.86	90	23.02	1189.89	0.57	38	31.31
p01-f40-v2t1	2403	97	24.77	1407.19	0.59	40	35.18
p01-f40-v2t2	-	-	-	7580.99	-	44	172.3
p01-l40-v1t1	305.09	84	3.63	185.2	0.61	40	4.63
p01-l40-v1t2	1633.48	108	15.12	1165.47	0.71	55	21.19
p01-l40-v2t1	2627.27	108	24.33	1386.55	0.53	44	31.51
p01-l40-v2t2	-	-	-	7560.21	-	48	157.5
p03-f40-v1t1	967.01	77	12.56	478.59	0.49	35	13.67
p03-f40-v1t2	7304.94	100	73.05	3020.07	0.41	45	67.11
p03-f40-v2t1	3665.82	124	29.56	1726.98	0.47	47	36.74
p03-f40-v2t2	-	-	-	1306.66	-	42	31.11
p03-m40-v1t1	372.55	89	4.19	192.03	0.52	37	5.19
p03-m40-v1t2	2102.19	117	17.97	976.05	0.46	44	22.18
p03-m40-v2t1	2080.48	101	20.6	879	0.42	45	19.53
p03-m40-v2t2	8504.13	112	75.93	638.1	0.08	38	16.79
p03-l40-v1t1	728.35	114	6.39	297.81	0.41	55	5.41
p03-l40-v1t2	2451.58	120	20.43	1622.2	0.66	64	25.35
p03-l40-v2t1	1898.16	92	20.63	929.9	0.49	43	21.63
p03-l40-v2t2	-	-	-	6838.52	-	44	155.42
p07-f40-v1t1	360.19	69	5.22	247.66	0.69	38	6.52
p07-f40-v1t2	1673.53	75	22.31	887	0.53	33	26.88
p07-f40-v2t1	2131.64	77	27.68	935.34	0.44	34	27.51
p07-f40-v2t2	11560.98	89	129.9	7571.9	0.65	45	168.26
p07-s40-v1t1	152.88	68	2.25	69.81	0.46	32	2.18
p07-s40-v1t2	580.9	76	7.64	220.19	0.38	28	7.86
p07-s40-v2t1	574.03	67	8.57	260.34	0.45	26	10.01
p07-s40-v2t2	3320.35	71	46.77	1877.25	0.57	34	55.21
p07-t40-v1t1	1026.73	77	13.33	534.15	0.52	36	14.84
p07-t40-v1t2	10006.02	80	125.08	928.37	0.09	38	24.43
p07-t40-v2t1	5200.24	76	68.42	3886.73	0.75	33	117.78
p07-t40-v2t2	-	-	-	-	-	-	-
Average	2819.46	90.37	30.91	1839.92	0.5	40.71	43.28

Table A.8: 1S-EBP with 2p-ESPPRC-LL and 2p-ESSPRC-CS: 25-customer instances

	1S-EBP with 2p-ESPPRC			1S-EBP with 2p-ESPPRC-LL + 2p-ESPPRC			
Data File	Tot.time (s) (a)	# of BP Nd.	Avg. CG time (s)	Tot.time (s) (b)	# of BP Nd.	Avg. CG time (s)	(b/a)
p01-f25-v1t1	80.86	30	2.58	77.81	25	2.99	0.96
p01-f25-v1t2	14866.19	199	74.19	12488.94	297	41.32	0.84
p01-f25-v2t1	833.58	95	8.67	594.65	99	5.87	0.71
p01-f25-v2t2	1639.84	151	11.06	954.41	119	7.79	0.58
p01-l25-v1t1	36.64	24	1.48	35.45	24	1.43	0.97
p01-l25-v1t2	218.77	201	0.97	173.39	139	1.14	0.79
p01-l25-v2t1	310.93	141	2.08	341.98	157	2.03	1.1
p01-l25-v2t2	527.24	37	14.41	414.51	39	10.48	0.79
p03-f25-v1t1	168.4	282	0.42	182.13	236	0.6	1.08
p03-f25-v1t2	140.62	69	1.89	147.16	77	1.77	1.05
p03-f25-v2t1	334.41	489	0.43	278.07	393	0.5	0.83
p03-f25-v2t2	314.08	49	6.58	316.88	52	5.92	1.01
p03-m25-v1t1	288.67	519	0.18	314.12	509	0.22	1.09
p03-m25-v1t2	x	x	x	x	x	x	x
p03-m25-v2t1	165.69	59	2.7	175.58	59	2.87	1.06
p03-m25-v2t2	835.65	9	93.19	523.75	9	57.89	0.63
p03-l25-v1t1	35.35	17	2.02	36.21	17	2.07	1.02
p03-l25-v1t2	106.14	37	2.8	113.9	37	3	1.07
p03-l25-v2t1	168.53	41	4.05	152.39	41	3.66	0.9
p03-l25-v2t2	564.31	41	13.86	452.17	37	12.11	0.8
p07-f25-v1t1	79.43	49	1.58	81.28	55	1.44	1.02
p07-f25-v1t2	6964.47	67	103.82	7874.52	67	117.4	1.13
p07-f25-v2t1	478.86	165	2.84	529.59	229	2.23	1.11
p07-f25-v2t2	1193.31	135	8.99	907.33	197	4.39	0.76
p07-s25-v1t1	63.61	499	0.08	87.77	599	0.1	1.38
p07-s25-v1t2	196.13	115	1.54	176.64	123	1.28	0.9
p07-s25-v2t1	30.92	21	1.42	33.88	21	1.57	1.1
p07-s25-v2t2	209.55	102	2.13	217.97	100	2.1	1.04
p07-t25-v1t1	73.65	35	2.02	51.8	35	1.4	0.7
p07-t25-v1t2	1258.27	1151	0.39	1498.69	1345	0.37	1.19
p07-t25-v2t1	1396.96	1011	0.51	1129.81	777	0.81	0.81
p07-t25-v2t2	789.76	87	9.39	754.9	91	7.92	0.96
Average	1108.74	191.19	12.2	1003.8	193.71	9.83	0.95

Table A.9: 1S-EBP with 2p-ESPPRC-LL and 2p-ESSPRC-CS: 25-customer instances, Cont.

	2p-ESPPRC-CS, 2p-ESPPRC				2p-ESPPRC-LL, 2p-ESPPRC-CS, 2p-ESPPRC			
Data File	Tot.time (s) (c)	# of BP Nd.	Avg. CG time (s)	(c/a)	Tot.time (s) (d)	# of BP Nd.	Avg. CG time (s)	(d/a)
p01-f25-v1t1	78.91	25	3.01	0.98	54.54	31	1.68	0.67
p01-f25-v1t2	11159.51	221	49.85	0.75	16832.67	197	84.85	1.13
p01-f25-v2t1	1815.81	79	22.85	2.18	1462.03	97	14.9	1.75
p01-f25-v2t2	897.41	101	8.68	0.55	1224.16	71	17.02	0.75
p01-l25-v1t1	40.07	26	1.48	1.09	42.03	22	1.84	1.15
p01-l25-v1t2	242.72	171	1.26	1.11	200.56	121	1.54	0.92
p01-l25-v2t1	380.64	155	2.3	1.22	347.62	157	2.06	1.12
p01-l25-v2t2	546.99	37	14.63	1.04	513.4	39	13.01	0.97
p03-f25-v1t1	167.02	256	0.46	0.99	162.08	268	0.42	0.96
p03-f25-v1t2	144.84	69	1.93	1.03	134.75	75	1.64	0.96
p03-f25-v2t1	212.76	289	0.55	0.64	304.99	579	0.32	0.91
p03-f25-v2t2	365.63	50	7.1	1.16	403.99	49	8.05	1.29
p03-m25-v1t1	417.63	659	0.16	1.45	341.56	567	0.19	1.18
p03-m25-v1t2	x	x	x	x	x	x	x	x
p03-m25-v2t1	216.57	63	3.31	1.31	213.6	59	3.49	1.29
p03-m25-v2t2	576.48	9	63.71	0.69	636.99	9	70.47	0.76
p03-l25-v1t1	34.43	17	1.97	0.97	36.6	17	2.08	1.04
p03-l25-v1t2	126.62	37	3.31	1.19	111.5	35	3.09	1.05
p03-l25-v2t1	185.95	47	3.89	1.1	170.48	43	3.91	1.01
p03-l25-v2t2	552.7	39	14.06	0.98	692.1	45	15.27	1.23
p07-f25-v1t1	90.86	59	1.48	1.14	95.54	57	1.63	1.2
p07-f25-v1t2	17030.53	71	239.67	2.45	7838.97	71	110.28	1.13
p07-f25-v2t1	485.99	205	2.3	1.01	571.31	263	2.1	1.19
p07-f25-v2t2	1076.36	219	4.71	0.9	1435.43	181	7.66	1.2
p07-s25-v1t1	49.33	291	0.13	0.78	65.99	441	0.11	1.04
p07-s25-v1t2	223.36	163	1.15	1.14	191.58	161	0.98	0.98
p07-s25-v2t1	36.5	21	1.68	1.18	32.91	21	1.51	1.06
p07-s25-v2t2	223.8	94	2.3	1.07	239.23	105	2.2	1.14
p07-t25-v1t1	69.15	35	1.89	0.94	76.94	39	1.88	1.04
p07-t25-v1t2	4061.29	2289	0.32	3.23	5506.6	2475	0.29	4.38
p07-t25-v2t1	1157.31	823	0.66	0.83	988.76	781	0.62	0.71
p07-t25-v2t2	1277.6	107	11.58	1.62	1244.91	149	7.91	1.58
Average	1417.57	217	15.24	1.18	1360.45	233.06	12.36	1.19

Table A.10: Performance with 2p-ESPPRC-LL and 2p-ESSPRC-CS: 40-customer instances

	1S-EBP with 2p-ESPPRC				1S-EBP with 2p-ESPPRC-LL + 2p-ESPPRC				
Data File	Tot.time (s) (a)	# of BP Nd.	Avg. CG time (s)	Gap %	Tot.time (s) (b)	# of BP Nd.	Avg. CG time (s)	Gap %	(b/a)
p01-f40-v1t1	19186.19	777	22.94	0	19060.3	935	18.17	0	0.99
p01-f40-v1t2	x	x	x	0.08	x	x	x	0.04	x
p01-f40-v2t1	12248.44	271	43.98	0	8910.97	262	32.77	0	0.73
p01-f40-v2t2	28808.63	12	2399.6	4.35	28800	*	*	4.28	1
p01-l40-v1t1	29133.11	67	434.37	6.79	28800	*	*	6.74	0.99
p01-l40-v1t2	10338.9	47	219.36	0	6701.66	47	142.07	0	0.65
p01-l40-v2t1	10006.77	91	109.51	0	8775.04	159	54.77	0	0.88
p01-l40-v2t2	28898.03	13	2221.55	7.9	18641.02	25	744.9	0	0.65
p03-f40-v1t1	28800	*	*	0.01	26697.32	83	321.32	0	0.93
p03-f40-v1t2	23429.75	39	600.35	0	25008.13	40	624.79	0	1.07
p03-f40-v2t1	3048.36	3	1015.07	0	2176.3	3	724.45	0	0.71
p03-f40-v2t2	28876.5	2	14435.53	6.19	28800	*	*	5.98	1
p03-m40-v1t1	18977.36	2423	2.33	0	16698.08	2407	1.44	0	0.88
p03-m40-v1t2	29199.38	311	90.15	0.06	28802.28	303	91.56	6.26	0.99
p03-m40-v2t1	28813.25	422	64.58	6.6	28800	*	*	6.61	1
p03-m40-v2t2	29053.1	16	1814.3	9.12	28800	*	*	8.38	0.99
p03-l40-v1t1	6419.83	203	30.93	0	5319.67	202	25.74	0	0.83
p03-l40-v1t2	28800.61	3646	5.06	0.03	28800	*	*	0.08	1
p03-l40-v2t1	10177.31	163	61.72	0	10255.35	167	60.59	0	1.01
p03-l40-v2t2	29018.43	13	2230.68	5.63	28837.69	25	1152.22	5.5	0.99
p07-f40-v1t1	2293.58	103	21.94	0	1891.68	103	18.03	0	0.82
p07-f40-v1t2	16511	741	21.42	0	10654.8	303	34.39	0	0.65
p07-f40-v2t1	22830.17	443	50.58	0	12299.16	457	26.11	0	0.54
p07-f40-v2t2	28830.47	18	1600.46	4.67	28821.53	30	959.71	4.56	1
p07-s40-v1t1	469.4	77	5.91	0	440.66	94	4.53	0	0.94
p07-s40-v1t2	886.68	23	38.27	0	618.44	23	26.63	0	0.7
p07-s40-v2t1	14276.36	265	53.15	0	5420.89	239	21.94	0	0.38
p07-s40-v2t2	22036.13	279	78.44	0	15782.07	207	75.79	0	0.72
p07-t40-v1t1	28800	*	*	0.16	28840.23	124	232.07	0.27	1
p07-t40-v1t2	28803.76	21	1370.22	2.71	28859.14	42	686.13	2.15	1
p07-t40-v2t1	28800	*	*	4.07	28800	*	*	4.07	1
p07-t40-v2t2	28000	*	*	**	28800	*	*	5.8	1.03
Average	20186.18	388.48	1075.64	1.88	18384.27	273.04	264.35	1.9	0.87

Table A.11: 1S-EBP with 2p-ESPPRC-LL and 2p-ESSPRC-CS: 40-customer instances, Cont.

	2p-ESPPRC-CS, 2p-ESPPRC					2p-ESPPRC-LL, 2p-ESPPRC-CS, 2p-ESPPRC				
Data File	Tot.time (s) (c)	# of BP Nd.	Avg. CG time (s)	Gap %	(c/a)	Tot.time (s) (d)	# of BP Nd.	Avg. CG time (s)	Gap %	(d/a)
p01-f40-v1t1	26943.98	773	32.75	0	1.4	28836.2	1082	24.04	0.01	1.5
p01-f40-v1t2	-	-	-	0.07	-	-	-	-	0.24	-
p01-f40-v2t1	11681.88	163	70.54	0	0.95	10523.39	241	42.28	0	0.86
p01-f40-v2t2	28800	12	*	4.35	1	28800	15	*	4.34	1
p01-l40-v1t1	28800	85	*	3.22	0.99	28858.46	160	179.34	6.32	0.99
p01-l40-v1t2	8577.94	44	194.35	0	0.83	7130.93	49	144.79	0	0.69
p01-l40-v2t1	12299.81	107	114.48	0	1.23	6978.62	77	90.16	0	0.7
p01-l40-v2t2	28800	16	*	7.88	1	22797.46	25	910.81	0	0.79
p03-f40-v1t1	28800	49	*	0.01	1	28800	66	*	0.01	1
p03-f40-v1t2	23798.86	39	609.81	0	1.02	26705.83	40	667.15	0	1.14
p03-f40-v2t1	2631.92	3	876.09	0	0.86	909.05	3	302.29	0	0.3
p03-f40-v2t2	28800	3	0	6.19	1	28842.86	6	4805.7	6.07	1
p03-m40-v1t1	24726.6	2499	3.44	0	1.3	20536.5	2395	2.09	0	1.08
p03-m40-v1t2	28800	301	*	4.27	0.99	28800	334	*	3.52	0.99
p03-m40-v2t1	28800	383	*	6.61	1	28800	451	*	6.59	1
p03-m40-v2t2	28800	20	*	8.96	0.99	28810.06	23	1251.14	8.57	0.99
p03-l40-v1t1	6309.74	198	31.22	0	0.98	5328.91	197	26.35	0	0.83
p03-l40-v1t2	28800	1735	*	0.08	1	-	-	-	0.08	0
p03-l40-v2t1	10110.99	165	60.63	0	0.99	8448.23	163	51.11	0	0.83
p03-l40-v2t2	28852.68	17	1696.11	5.58	0.99	28800	22	*	5.53	0.99
p07-f40-v1t1	2122.83	103	20.29	0	0.93	1940.71	103	18.51	0	0.85
p07-f40-v1t2	19084.75	497	37.56	0	1.16	7999.36	261	29.98	0	0.48
p07-f40-v2t1	17727.98	357	48.76	0	0.78	11036.22	419	25.58	0	0.48
p07-f40-v2t2	28871.14	27	1068.45	4.62	1	28800	32	*	4.55	1
p07-s40-v1t1	514.74	98	5.05	0	1.1	424.88	83	4.92	0	0.91
p07-s40-v1t2	924.37	23	39.87	0	1.04	544.78	23	23.41	0	0.61
p07-s40-v2t1	7630.35	255	29.18	0	0.53	4136.89	247	16.14	0	0.29
p07-s40-v2t2	18944.23	231	81.53	0	0.86	13794.52	269	50.83	0	0.63
p07-t40-v1t1	28800	152	*	0.16	1	29211.47	162	179.58	0.07	1.01
p07-t40-v1t2	28800	15	*	3.04	1	28832.01	53	543.16	1.84	1
p07-t40-v2t1	28800	18	*	4.07	1	28800	18	*	4.07	1
p07-t40-v2t2	28800	3	*	5.9	1.03	28800	5	*	5.8	1.03
Average	20198.54	270.68	264.22	2.03	1	18434.24	234.13	426.79	1.8	0.84

Table A.12: HBP with 2p-ESPPRC-LL and 2p-ESSPRC-CS

		2p-ESPPRC-LL			2p-ESPPRC-LL, 2p-ESPPRC-CS-12			2p-ESPPRC-LL, 2p-ESPPRC-CS-20		
Data File	Best IP	UB	Gap %	time (s)	UB	Gap %	time (s)	UB	Gap %	time (s)
p01-f40-v1t1	7131	7131	0	1594.47	7131	0	4635.03	7131	0	3191.8
p01-f40-v1t2	6983	6984	0.01	7201.8	6983	0	-	7170	2.68	7202.76
p01-f40-v2t1	6866	6866	0	203.18	6866	0	396.65	6866	0	293.74
p01-f40-v2t2	6818	6818	0	7202.26	6819	0.01	7205.48	6821	0.04	2123.59
p01-l40-v1t1	7159	7159	0	7202.5	7214	0.77	7207.93	7183	0.34	7206.03
p01-l40-v1t2	6934	6934	0	144.57	6934	0	292.24	6934	0	333.1
p01-l40-v2t1	6823	6823	0	51.64	6823	0	141.3	6823	0	145.77
p01-l40-v2t2	6612	6613	0.02	98.7	6621	0.14	339	6633	0.32	506.36
p03-f40-v1t1	8780	8780	0	270.95	8780	0	537.47	8780	0	574.53
p03-f40-v1t2	8753	8753	0	1509.47	8753	0	1559.74	8753	0	1679.57
p03-f40-v2t1	8440	8663	2.64	237.29	8663	2.64	446.09	8663	2.64	467.15
p03-f40-v2t2	8438	8441	0.04	134.27	8439	0.01	872.07	8438	0	690.57
p03-m40-v1t1	7253	7755	6.92	7204.8	7257	0.06	7201.92	7740	6.71	7200.88
p03-m40-v1t2	7075	7075	0	7238.97	7075	0	7205.53	7236	2.28	5037.24
p03-m40-v2t1	7003	7132	1.84	6900.02	7003	0	4211.66	7132	1.84	7202.01
p03-m40-v2t2	6947	6967	0.29	7202.03	6947	0	2436.43	6947	0	2234.61
p03-l40-v1t1	7167	7167	0	160.1	7167	0	223.06	7167	0	90.81
p03-l40-v1t2	6950	6950	0	7201.62	6950	0	7202.16	6950	0	7200.81
p03-l40-v2t1	6849	6849	0	130.85	6849	0	190.73	6849	0	232.33
p03-l40-v2t2	6639	6846	3.12	435.09	6846	3.12	535.26	6846	3.12	2341.28
p07-f40-v1t1	7167	7167	0	62.23	7167	0	172.17	7167	0	195.24
p07-f40-v1t2	7001	7001	0	506.83	7001	0	525.29	7001	0	662.4
p07-f40-v2t1	6881	6905	0.35	52.96	6881	0	485.49	6881	0	510.22
p07-f40-v2t2	6845	6850	0.07	762.36	6845	0	1235.29	6845	0	1218.72
p07-s40-v1t1	7343	7343	0	110.02	7343	0	46.67	7343	0	45.76
p07-s40-v1t2	7117	7117	0	37.54	7124	0.1	47.73	7124	0.1	107.18
p07-s40-v2t1	7000	7008	0.11	213.42	7000	0	62.46	7005	0.07	212.83
p07-s40-v2t2	6944	6944	0	444.3	6945	0.01	375.46	6944	0	341.39
p07-t40-v1t1	7012	7015	0.04	796.04	7012	0	504.59	7013	0.01	613.31
p07-t40-v1t2	6972	6972	0	7203.88	6972	0	7206.91	6972	0	4572.11
p07-t40-v2t1	6665	6858	2.9	1099.94	6858	2.9	2239.16	6857	2.88	4090.15
p07-t40-v2t2	6658	6658	0	3465.59	6660	0.03	822.08	6658	0	1566.81
Average			0.57	2408.74		0.31	2147.2		0.72	2190.35

Table A.13: 2S-EBP with different designs of HBP

	2S-EBP With HBP that uses								
	no 2p-ESPPRC-CS			2p-ESPPRC-CS-12			2p-ESPPRC-CS-20		
Data	Gap	Step 2 (s)	Tot (s)	Gap	Step 2 (s)	Tot (s)	Gap	Step 2 (s)	Tot (s)
p01-f40-v1t1	0.06	21600.52	23194.99	0	17884.51	22519.54	0	9002.07	12193.87
p01-f40-v1t2	0.33	21603.02	28804.82	0	-	-	0.32	21600	28802.76
p01-f40-v2t1	0	11176.92	11380.1	0	10823.79	11220.44	0	6779.42	7073.16
p01-f40-v2t2	2.03	21780.84	28983.1	2.04	21767.73	28973.21	1.26	21640.84	23764.43
p01-l40-v1t1	3.02	21601.15	28803.65	3.71	21684.45	28892.38	3.22	21622.19	28828.22
p01-l40-v1t2	0	6330.69	6475.26	0	5239.58	5531.82	0	2880.06	3213.16
p01-l40-v2t1	0	9989.3	10040.94	0	9238.55	9379.85	0	6191.15	6336.92
p01-l40-v2t2	0.01	21691.61	21790.31	0	19555.27	19894.27	0	11253.68	11760.04
p03-f40-v1t1	0	9241.16	9512.11	0	21890.4	22427.87	0	5868.19	6442.72
p03-f40-v1t2	0	21572.76	23082.23	0	18963.12	20522.86	0	9276.91	10956.48
p03-f40-v2t1	0	991.78	1229.07	0	1016.06	1462.15	0	528.56	995.71
p03-f40-v2t2	0.49	21606.8	21741.07	0.38	21645.1	22517.17	0	8293.47	8984.04
p03-m40-v1t1	0.41	21610.1	28814.9	0.08	21600.67	28802.59	0	21604.08	28804.96
p03-m40-v1t2	0.08	21702.66	28941.63	0.08	21884.13	29089.66	2.95	2186.5	7223.74
p03-m40-v2t1	2.14	21626.98	28527	0.33	21604.12	25815.78	2.79	2163.77	9365.78
p03-m40-v2t2	3.01	21682.05	28884.08	2.59	22169.5	24605.93	4.18	2175.46	4410.07
p03-l40-v1t1	0	5515.75	5675.85	0	5640.92	5863.98	0	3334.91	3425.72
p03-l40-v1t2	0.05	21607.14	28808.76	0.04	21600.13	28802.29	0.05	21600	28800.81
p03-l40-v2t1	0	10199.23	10330.08	0	10642.46	10833.19	0	6828.42	7060.75
p03-l40-v2t2	3.14	21607.06	22042.15	3.2	21604.89	22140.15	0	17453.7	19794.98
p07-f40-v1t1	0	1927.39	1989.62	0	1824.84	1997.01	0	1325.01	1520.25
p07-f40-v1t2	0	7798.67	8305.5	0	10076.08	10601.37	0	4527.12	5189.52
p07-f40-v2t1	0.01	23958.09	24011.05	0	12047.83	12533.32	0	8941.95	9452.17
p07-f40-v2t2	2.53	21623.87	22386.23	2.46	21651.7	22886.99	2.34	21607.55	22826.27
p07-s40-v1t1	0	326.04	436.06	0	274.13	320.8	0	263.38	309.14
p07-s40-v1t2	0	516.54	554.08	0	662.59	710.32	0	390.17	497.35
p07-s40-v2t1	0	11463.4	11676.82	0	7868.75	7931.21	0	7716.19	7929.02
p07-s40-v2t2	0	16099.52	16543.82	0	14168.82	14544.28	0	12673.15	13014.54
p07-t40-v1t1	0.07	22408.12	23204.16	0.03	21614.97	22119.56	0	8870.59	9483.9
p07-t40-v1t2	0.94	21922.1	29125.98	1.21	21608.45	28815.36	0.35	21627.34	26199.45
p07-t40-v2t1	2.85	21700	22799.94	2.85	22600	24839.16	0.04	23464.62	27554.77
p07-t40-v2t2	1.72	22390.12	25855.71	1.76	21764.63	22586.71	1.64	21800.96	23367.77
Average	0.72	15839.73	18248.47	0.65	15245.75	17392.94	0.6	10484.11	12674.45

Table A.14: Performance of I-Heur, BB, HBP: 25 customer instances

Data	Upper Bound Values				Time (s)			
	I-Heur	BB in HBP Rt	HBP W BB	HBP WO BB	I-Heur	BB in HBP Rt	HBP W BB	HBP WO BB
p01-f25-v1t1	4795	4722	4699	4699	0.02	10.5	18.32	9.26
p01-f25-v1t2	4795	4734	4511	4511	0.05	40.54	166.61	223.87
p01-f25-v2t1	4722	4701	4427	4425	0.02	27.7	31.09	19.49
p01-f25-v2t2	4722	4710	4425	4425	0.04	34.41	55.99	53.75
p01-l25-v1t1	5193	4963	4899	4947	0.04	2.59	10.56	12.87
p01-l25-v1t2	4968	4827	4815	4815	0.03	3.54	17.66	44.54
p01-l25-v2t1	4809	4785	4759	4759	0.04	0.87	13.82	21.26
p01-l25-v2t2	4762	4762	4650	4650	0.04	1.7	9.61	9.19
p03-f25-v1t1	5281	5082	5062	5062	0.06	6.06	67.16	71.24
p03-f25-v1t2	5139	5086	4837	4837	0.04	65.71	96.88	33.25
p03-f25-v2t1	5048	4785	4759	4759	0.04	1.54	73.3	621.59
p03-f25-v2t2	4831	4831	4742	4742	0.04	19.04	34.77	19.04
p03-m25-v1t1	4995	4853	4785	4785	0.04	28.28	101.27	208
p03-m25-v1t2	4869	4801	4785	4785	0.05	47.89	7200	7200
p03-m25-v2t1	4992	4726	4689	4689	0.04	4.56	21.79	18.63
p03-m25-v2t2	4767	4761	4548	4548	0.03	11.55	26.04	18.11
p03-l25-v1t1	5186	5073	4921	4921	0.04	5.31	10.34	8.7
p03-l25-v1t2	4871	4848	4803	4803	0.03	6.73	13.29	6.87
p03-l25-v2t1	4805	4805	4741	4741	0.02	1.28	6.65	6.34
p03-l25-v2t2	4805	4794	4734	4734	0.03	3.4	18.51	19.98
p07-f25-v1t1	4870	4768	4761	4761	0.03	1.51	9.34	8.67
p07-f25-v1t2	4823	4791	4761	4761	0.04	8.92	37.16	23.87
p07-f25-v2t1	4721	4721	4703	4703	0.02	1.96	72.08	40.57
p07-f25-v2t2	4721	4576	4482	4482	0.03	0.92	24.08	49.14
p07-s25-v1t1	5412	5103	5098	5098	0.04	1.34	17.34	52.67
p07-s25-v1t2	5238	5049	4889	4889	0.04	7.89	25.86	17.88
p07-s25-v2t1	5159	5030	4787	4787	0.04	8.57	11.8	3.88
p07-s25-v2t2	4934	4856	4781	4781	0.04	5.38	21.35	18.4
p07-t25-v1t1	5134	5128	4898	4898	0.03	8.1	15.01	7.63
p07-t25-v1t2	5162	5121	4886	4886	0.04	35.68	373.5	1690.54
p07-t25-v2t1	5025	4800	4771	4767	0.04	1.32	144.38	710.75
p07-t25-v2t2	4769	4769	4654	4607	0.03	3.4	14.76	13.25
Average					0.04	12.76	273.76	351.98

Table A.15: Performance of I-Heur, BB, HBP: 40-customer instances

Data	Upper Bound Values				Time (s)			
	I-Heur	BB in HBP Rt	HBP W BB	HBP WO BB	I-Heur	BB in HBP Rt	HBP W BB	HBP WO BB
p01-f40-v1t1	7170	7170	7131	7131	0.05	10.7	3191.8	3128.83
p01-f40-v1t2	7170	7170	7170	7170	0.04	241.72	7202.76	7206.14
p01-f40-v2t1	7064	7064	6866	6866	0.06	20.32	293.74	286.94
p01-f40-v2t2	6984	6984	6821	6821	0.05	48.86	2123.59	2049.08
p01-l40-v1t1	7449	7339	7183	7179	0.07	218.09	7206.03	7206.02
p01-l40-v1t2	7236	7199	6934	6934	0.06	128.43	333.1	223.53
p01-l40-v2t1	7166	6970	6823	6823	0.04	9.21	145.77	149.28
p01-l40-v2t2	7166	7133	6633	6638	0.07	86.56	506.36	763.49
p03-f40-v1t1	9253	9013	8780	8780	0.04	45.13	574.53	722.31
p03-f40-v1t2	9028	8760	8753	8753	0.04	218.16	1679.57	1535.31
p03-f40-v2t1	8943	8922	8663	8663	0.06	98.67	467.15	353.33
p03-f40-v2t2	8943	8702	8438	8438	0.04	50.65	690.57	659.91
p03-m40-v1t1	7827	7740	7740	7827	0.06	2303.69	7200.88	7203.83
p03-m40-v1t2	7544	7503	7236	7236	0.1	3637.17	5037.24	2286.2
p03-m40-v2t1	7479	7479	7132	7132	0.06	449.2	7202.01	7200
p03-m40-v2t2	7430	7430	6947	6947	0.11	1339	2234.61	940.01
p03-l40-v1t1	7386	7193	7167	7167	0.05	10.42	90.81	95.16
p03-l40-v1t2	7212	7212	6950	6950	0.05	72.29	7200.81	7201.88
p03-l40-v2t1	7121	7112	6849	6849	0.05	16.84	232.33	209.03
p03-l40-v2t2	7018	7018	6846	6846	0.05	187.95	2341.28	2226.85
p07-f40-v1t1	7424	7424	7167	7167	0.04	86.14	195.24	117.71
p07-f40-v1t2	7340	7246	7001	7001	0.05	78.8	662.4	684.56
p07-f40-v2t1	7174	7173	6881	6881	0.05	25.72	510.22	677.89
p07-f40-v2t2	7000	6942	6845	6709	0.05	36.09	1218.72	1025.11
p07-s40-v1t1	7860	7367	7343	7343	0.05	4.92	45.76	46.34
p07-s40-v1t2	7656	7372	7124	7124	0.05	39.11	107.18	87.86
p07-s40-v2t1	7466	7441	7005	7005	0.04	137.35	212.83	87.12
p07-s40-v2t2	7241	7001	6944	6944	0.05	14.8	341.39	503.35
p07-t40-v1t1	7283	7283	7013	7012	0.05	12.78	613.31	555.03
p07-t40-v1t2	7058	7058	6972	6972	0.05	20.21	4572.11	4536.15
p07-t40-v2t1	6945	6945	6857	6857	0.04	15.51	4090.15	1401.86
p07-t40-v2t2	6945	6945	6658	6669	0.04	83.95	1566.81	2277.89
Average					0.05	304.64	2190.35	1989

Table A.16: I-Heur, BB, HBPs Upper Bounds vs. LP Relaxations & Optimal (Best) IPs: 25-customer

	%Gap btw UB & Optimal (or Best IP)				%Gap btw UB & the LP Relax.			
Data	I-Heur	BB in HBP Rt	HBP W BB	HBP WO BB	I-Heur	BB in HBP Rt	HBP W BB	HBP WO BB
p01-f25-v1t1	2.04	0.49	0	0	6.45	4.83	4.32	4.32
p01-f25-v1t2	6.3	4.94	0	0	8.34	6.96	1.93	1.93
p01-f25-v2t1	6.71	6.24	0.05	0	8.15	7.67	1.39	1.35
p01-f25-v2t2	6.71	6.44	0	0	9.93	9.65	3.02	3.02
p01-l25-v1t1	6	1.31	0	0.98	7.08	2.33	1.01	2
p01-l25-v1t2	3.18	0.25	0	0	5.13	2.15	1.9	1.9
p01-l25-v2t1	1.05	0.55	0	0	3	2.49	1.93	1.93
p01-l25-v2t2	3.48	3.48	1.04	1.04	4.96	4.96	2.49	2.49
p03-f25-v1t1	4.33	0.4	0	0	9.79	5.66	5.24	5.24
p03-f25-v1t2	6.24	5.15	0	0	9.31	8.19	2.89	2.89
p03-f25-v2t1	6.09	0.57	0.02	0.02	9.59	3.88	3.32	3.32
p03-f25-v2t2	1.88	1.88	0	0	7.29	7.29	5.32	5.32
p03-m25-v1t1	4.39	1.42	0	0	6.34	3.31	1.87	1.87
p03-m25-v1t2	1.76	0.33	0	0	6.35	4.87	4.52	4.52
p03-m25-v2t1	6.46	0.79	0	0	11.28	5.35	4.53	4.53
p03-m25-v2t2	6.43	6.3	1.54	1.54	8.44	8.3	3.45	3.45
p03-l25-v1t1	7.1	4.77	1.63	1.63	8.15	5.79	2.62	2.62
p03-l25-v1t2	1.42	0.94	0	0	4.3	3.81	2.84	2.84
p03-l25-v2t1	1.35	1.35	0	0	3.98	3.98	2.6	2.6
p03-l25-v2t2	3.22	2.99	1.7	1.7	6.6	6.35	5.02	5.02
p07-f25-v1t1	2.29	0.15	0	0	4.9	2.7	2.55	2.55
p07-f25-v1t2	2.95	2.26	1.62	1.62	6.13	5.43	4.77	4.77
p07-f25-v2t1	0.38	0.38	0	0	5.25	5.25	4.85	4.85
p07-f25-v2t2	5.33	2.1	0	0	7.58	4.28	2.13	2.13
p07-s25-v1t1	6.16	0.1	0	0	8.24	2.06	1.96	1.96
p07-s25-v1t2	7.14	3.27	0	0	7.95	4.06	0.76	0.76
p07-s25-v2t1	7.77	5.08	0	0	8.85	6.13	1	1
p07-s25-v2t2	3.2	1.57	0	0	7.03	5.34	3.71	3.71
p07-t25-v1t1	4.82	4.7	0	0	6.15	6.03	1.27	1.27
p07-t25-v1t2	5.65	4.81	0	0	9.76	8.89	3.89	3.89
p07-t25-v2t1	5.41	0.69	0.08	0	8.15	3.31	2.69	2.6
p07-t25-v2t2	3.85	3.85	1.35	0.33	5.67	5.67	3.12	2.08
Average	4.41	2.48	0.28	0.28	7.19	5.22	2.97	2.96

Table A.17: I-Heur, BB, HBPs Upper Bounds vs. LP Relaxations & Optimal (Best) IPs: 40-customer

	%Gap btw UB & Optimal (or Best IP)				%Gap btw UB & the LP Relax.			
Data	I-Heur	BB in HBP Rt	HBP W BB	HBP WO BB	I-Heur	BB in HBP Rt	HBP W BB	HBP WO BB
p01-f40-v1t1	0.55	0.55	0	0	2.78	2.78	2.22	2.22
p01-f40-v1t2	2.68	2.68	2.68	2.68	5.04	5.04	5.04	5.04
p01-f40-v2t1	2.88	2.88	0	0	5.43	5.43	2.47	2.47
p01-f40-v2t2	2.43	2.43	0.04	0.04	6.12	6.12	3.65	3.65
p01-l40-v1t1	4.05	2.51	0.34	0.28	7.96	6.37	4.11	4.05
p01-l40-v1t2	4.36	3.82	0	0	7.08	6.54	2.61	2.61
p01-l40-v2t1	5.03	2.15	0	0	8.3	5.34	3.12	3.12
p01-l40-v2t2	8.38	7.88	0.32	0.39	10.02	9.51	1.83	1.91
p03-f40-v1t1	5.39	2.65	0	0	6.39	3.63	0.95	0.95
p03-f40-v1t2	3.14	0.08	0	0	5.24	2.11	2.03	2.03
p03-f40-v2t1	5.96	5.71	2.64	2.64	5.97	5.72	2.65	2.65
p03-f40-v2t2	5.98	3.13	0	0	6.59	3.71	0.57	0.57
p03-m40-v1t1	7.91	6.71	6.71	7.91	11.01	9.77	9.77	11.01
p03-m40-v1t2	6.63	6.05	2.28	2.28	9.21	8.62	4.75	4.75
p03-m40-v2t1	6.8	6.8	1.84	1.84	10.08	10.08	4.98	4.98
p03-m40-v2t2	6.95	6.95	0	0	11.64	11.64	4.38	4.38
p03-l40-v1t1	3.06	0.36	0	0	6.73	3.94	3.56	3.56
p03-l40-v1t2	3.77	3.77	0	0	6.5	6.5	2.64	2.64
p03-l40-v2t1	3.97	3.84	0	0	6.88	6.74	2.79	2.79
p03-l40-v2t2	5.71	5.71	3.12	3.12	7.23	7.23	4.6	4.6
p07-f40-v1t1	3.59	3.59	0	0	5.26	5.26	1.62	1.62
p07-f40-v1t2	4.84	3.5	0	0	6.28	4.92	1.37	1.37
p07-f40-v2t1	4.26	4.24	0	0	6.35	6.34	2.01	2.01
p07-f40-v2t2	4.34	3.47	2.03	0	5.59	4.72	3.25	1.2
p07-s40-v1t1	7.04	0.33	0	0	8.32	1.53	1.19	1.19
p07-s40-v1t2	7.57	3.58	0.1	0.1	7.97	3.96	0.46	0.46
p07-s40-v2t1	6.66	6.3	0.07	0.07	7.82	7.46	1.16	1.16
p07-s40-v2t2	4.28	0.82	0	0	7.07	3.52	2.68	2.68
p07-t40-v1t1	3.86	3.86	0.01	0	4.93	4.93	1.04	1.03
p07-t40-v1t2	1.23	1.23	0	0	3.69	3.69	2.43	2.43
p07-t40-v2t1	4.2	4.2	2.88	2.88	4.44	4.44	3.12	3.12
p07-t40-v2t2	4.31	4.31	0	0.17	6.27	6.27	1.88	2.04
Average	4.74	3.63	0.78	0.76	6.88	5.75	2.84	2.82

Table A.18: Performance of BB at Root Node of 1S-EBP: 25-customer

Data File	1S-EBP, BB at Root			1S-EBP WO BB		Time BB/WOBB
	BBIP-Opt Gap	Nds	Time	Nds	Time	
p01-f25-v1t1	0.19	25	95.11	25	77.81	1.22
p01-f25-v1t2	4.54	231	14091.47	297	12488.94	1.13
p01-f25-v2t1	0	91	697.91	99	594.65	1.17
p01-f25-v2t2	0.54	101	1233.77	119	954.41	1.29
p01-l25-v1t1	2.84	24	42.06	24	35.45	1.19
p01-l25-v1t2	0	123	191.48	139	173.39	1.1
p01-l25-v2t1	0.65	157	382.42	157	341.98	1.12
p01-l25-v2t2	1.3	39	491.26	39	414.51	1.19
p03-f25-v1t1	0.4	278	214.85	236	182.13	1.18
p03-f25-v1t2	0.74	81	187.65	77	147.16	1.28
p03-f25-v2t1	4.77	445	391.13	393	278.07	1.41
p03-f25-v2t2	0.72	48	393.62	52	316.88	1.24
p03-m25-v1t1	4.39	509	360.64	509	314.12	1.15
p03-m25-v1t2	0.4	-	-	-	-	-
p03-m25-v2t1	0.66	59	223.55	59	175.58	1.27
p03-m25-v2t2	6.43	9	620.86	9	523.75	1.19
p03-l25-v1t1	4.07	17	43.37	17	36.21	1.2
p03-l25-v1t2	0.5	37	118.88	37	113.9	1.04
p03-l25-v2t1	0.27	41	160.12	41	152.39	1.05
p03-l25-v2t2	2.34	37	519.68	37	452.17	1.15
p07-f25-v1t1	0	47	83.08	55	81.28	1.02
p07-f25-v1t2	1.66	67	10990.51	67	7874.52	1.4
p07-f25-v2t1	0.26	229	569.53	229	529.59	1.08
p07-f25-v2t2	0.29	185	1010.26	197	907.33	1.11
p07-s25-v1t1	0.1	501	79.39	599	87.77	0.9
p07-s25-v1t2	4.79	123	202.86	123	176.64	1.15
p07-s25-v2t1	0.08	21	35.61	21	33.88	1.05
p07-s25-v2t2	0	97	237.78	100	217.97	1.09
p07-t25-v1t1	0.1	41	66.95	35	51.8	1.29
p07-t25-v1t2	0.78	593	687.63	1345	1498.69	0.46
p07-t25-v2t1	0.08	957	1478.93	777	1129.81	1.31
p07-t25-v2t2	3.85	91	859.93	91	754.9	1.14
Average	1.49	171.1	1185.88	193.71	1003.8	1.15

Table A.19: Performance of BB at Root Node of 1S-EBP: 40-customer

Data File	1S-EBP, BB at Root				1S-EBP WO BB			Time
	BBIP-Opt Gap	Nds	Gap	Time	Nds	Gap	Time	BB/WOBB
p01-f40-v1t1	0.55	935	0	19701.81	935	0	19060.3	1.03
p01-f40-v1t2	2.68	1208	0.04	-	-	0.04	-	-
p01-f40-v2t1	2.04	262	0	12093.16	262	0	8910.97	1.36
p01-f40-v2t2	1.38	14	3.34	28842.48	*	4.28	28900	1
p01-l40-v1t1	0.28	174	3.24	28881.47	*	6.74	28900	1
p01-l40-v1t2	0.48	43	0	7159.8	47	0	6701.66	1.07
p01-l40-v2t1	0.13	143	0	9707.42	159	0	8775.04	1.11
p01-l40-v2t2	3.64	25	0	22005.02	25	0	18641.02	1.18
p03-f40-v1t1	2.32	58	0.01	28900	83	0	26697.32	1.08
p03-f40-v1t2	0	39	0	27551.15	40	0	25008.13	1.1
p03-f40-v2t1	2.67	3	0	2435.82	3	0	2176.3	1.12
p03-f40-v2t2	0.05	6	0.51	28840.91	*	5.98	28900	1
p03-m40-v1t1	3.65	2521	0	21399.49	2407	0	16698.08	1.28
p03-m40-v1t2	7.31	231	6.9	28825.64	303	6.26	28802.28	1
p03-m40-v2t1	*	1	*	28900	*	6.61	28900	1
p03-m40-v2t2	0	26	2.16	28832.76	*	8.38	28900	1
p03-l40-v1t1	0.43	199	0	5869.16	202	0	5319.67	1.1
p03-l40-v1t2	0.65	3381	0.08	-	*	0.08	28900	-
p03-l40-v2t1	3.59	167	0	12038.32	167	0	10255.35	1.17
p03-l40-v2t2	3.68	23	3.66	28812.64	25	5.5	28837.69	1
p07-f40-v1t1	1.14	101	0	1938.89	103	0	1891.68	1.02
p07-f40-v1t2	3.44	391	0	12329.6	303	0	10654.8	1.16
p07-f40-v2t1	3.07	529	0	16487.51	457	0	12299.16	1.34
p07-f40-v2t2	3.2	27	3.57	28831.65	30	4.56	28821.53	1
p07-s40-v1t1	2.89	94	0	535.39	94	0	440.66	1.21
p07-s40-v1t2	2.89	23	0	629.55	23	0	618.44	1.02
p07-s40-v2t1	2.81	247	0	8643.47	239	0	5420.89	1.59
p07-s40-v2t2	0.65	295	0	18052.95	207	0	15782.07	1.14
p07-t40-v1t1	3.14	156	0.07	28900	124	0.27	28840.23	1
p07-t40-v1t2	1.16	41	2.08	28854.13	42	2.15	28859.14	1
p07-t40-v2t1	3.66	18	3.57	28900	*	4.07	28900	1
p07-t40-v2t2	0.56	5	2.19	28823.02	*	5.8	28900	1
Average	2.07		1.05	19124.11		1.9	18410.08	1.1

Table A.20: 1S-EBP vs 2S-EBP: Effect of HBP, 25-Customer Instances

Data File	Nds		CPU Time (s)		Time
	1S-EBP	2S-EBP	1S-EBP	2S-EBP	1S-EBP/2S-EBP
p01-f25-v1t1	25	27	77.81	72.82	1.07
p01-f25-v1t2	297	125	12488.94	21876.01	0.57
p01-f25-v2t1	99	107	594.65	802.14	0.74
p01-f25-v2t2	119	159	954.41	1120.17	0.85
p01-l25-v1t1	24	25	35.45	34.65	1.02
p01-l25-v1t2	139	119	173.39	165.53	1.05
p01-l25-v2t1	157	145	341.98	329.88	1.04
p01-l25-v2t2	39	39	414.51	522.43	0.79
p03-f25-v1t1	236	235	182.13	241.56	0.75
p03-f25-v1t2	77	121	147.16	178.85	0.82
p03-f25-v2t1	393	743	278.07	603.42	0.46
p03-f25-v2t2	52	47	316.88	298.62	1.06
p03-m25-v1t1	509	357	314.12	340.02	0.92
p03-m25-v1t2	x	x	x	x	x
p03-m25-v2t1	59	59	175.58	180.23	0.97
p03-m25-v2t2	9	9	523.75	538.48	0.97
p03-l25-v1t1	17	17	36.21	36.8	0.98
p03-l25-v1t2	37	37	113.9	109.61	1.04
p03-l25-v2t1	41	51	152.39	161.1	0.95
p03-l25-v2t2	37	43	452.17	543.47	0.83
p07-f25-v1t1	55	37	81.28	79.15	1.03
p07-f25-v1t2	67	61	7874.52	13085.64	0.6
p07-f25-v2t1	229	181	529.59	466.23	1.14
p07-f25-v2t2	197	137	907.33	880.35	1.03
p07-s25-v1t1	599	305	87.77	65.79	1.33
p07-s25-v1t2	123	99	176.64	173.24	1.02
p07-s25-v2t1	21	23	33.88	41.57	0.82
p07-s25-v2t2	100	85	217.97	208.18	1.05
p07-t25-v1t1	35	47	51.8	96.07	0.54
p07-t25-v1t2	1345	3323	1498.69	10426.22	0.14
p07-t25-v2t1	777	1033	1129.81	2039.87	0.55
p07-t25-v2t2	91	127	754.9	1065.28	0.71
Average			1003.8	1831.72	0.87

Table A.21: 1S-EBP vs 2S-EBP: Effect of HBP, 40-customer instances

Data File	Gap %		Nds		CPU Time (s)		Time
	1S-EBP	2S-EBP	1S-EBP	2S-EBP	1S-EBP	2S-EBP	2S-EBP/2S-EBP
p01-f40-v1t1	0	0	935	1047	19060.3	24008.84	0.79
p01-f40-v1t2	0.04	0.14	x	1259	x	28805.21	x
p01-f40-v2t1	0	0	262	185	8910.97	9143.19	0.97
p01-f40-v2t2	4.28	1.99	*	15	28900	24308.9	1.19
p01-l40-v1t1	6.74	3.29	*	179	28900	29026.46	1
p01-l40-v1t2	0	0	47	43	6701.66	5697.16	1.18
p01-l40-v2t1	0	0	159	97	8775.04	8983.17	0.98
p01-l40-v2t2	0	0	25	25	18641.02	17638.15	1.06
p03-f40-v1t1	0	0.01	83	58	26697.32	22503.4	1.19
p03-f40-v1t2	0	0	40	39	25008.13	21788.5	1.15
p03-f40-v2t1	0	0	3	3	2176.3	1776.48	1.23
p03-f40-v2t2	5.98	0.46	*	5	28900	22358.28	1.29
p03-m40-v1t1	0	0.41	2407	2191	16698.08	28812.76	0.58
p03-m40-v1t2	6.26	2.34	303	182	28802.28	26668.97	1.08
p03-m40-v2t1	6.61	2.56	*	205	28900	28812.92	1
p03-m40-v2t2	8.38	2.59	*	21	28900	24508.88	1.18
p03-l40-v1t1	0	0	202	199	5319.67	5650.57	0.94
p03-l40-v1t2	0.08	0.04	*	1471	28900	28801.69	1
p03-l40-v2t1	0	0	167	161	10255.35	10288.34	1
p03-l40-v2t2	5.5	3.08	25	24	28837.69	24050.68	1.2
p07-f40-v1t1	0	0	103	105	1891.68	2034.98	0.93
p07-f40-v1t2	0	0	303	427	10654.8	6579.34	1.62
p07-f40-v2t1	0	0	457	243	12299.16	11315.02	1.09
p07-f40-v2t2	4.56	2.46	30	26	28821.53	22903.7	1.26
p07-s40-v1t1	0	0	94	77	440.66	393.97	1.12
p07-s40-v1t2	0	0	23	23	618.44	687.58	0.9
p07-s40-v2t1	0	0	239	245	5420.89	7570.69	0.72
p07-s40-v2t2	0	0	207	259	15782.07	17281.98	0.91
p07-t40-v1t1	0.27	0.03	124	168	28840.23	23413.31	1.23
p07-t40-v1t2	2.15	1.06	42	42	28859.14	26690.69	1.08
p07-t40-v2t1	4.07	0.07	*	18	28900	26890.15	1.07
p07-t40-v2t2	5.8	1.73	*	3	28900	23295.55	1.24
Average	1.9	0.7			18410.08	17584.05	1.07

Table A.22: Cover and GAP Inequalities for 25-customer Instances

	1S-EBP-WO	1S-EBP with CUTGEN$_G$			1S-EBP with CUTGEN$_K$		
Data file	Nds	Nds	Time W/WO	# cuts	Nds	Time W/WO	# cuts
p01-f25-v1t1	28	26	1.01	1	24	1.11	5
p01-f25-v1t2	81	87	1.05	1	87	0.98	2
p01-f25-v2t1	25	25	0.96	0	33	1.05	1
p01-f25-v2t2	29	29	1	0	41	0.97	3
p01-l25-v1t1	29	29	0.95	0	29	0.95	0
p01-l25-v1t2	69	69	0.99	0	71	1.02	1
p01-l25-v2t1	127	127	0.98	0	132	0.94	5
p01-l25-v2t2	33	33	0.96	1	33	0.99	1
p03-f25-v1t1	241	233	1.04	9	189	0.88	32
p03-f25-v1t2	60	60	0.99	5	56	1.27	20
p03-f25-v2t1	151	193	1.03	2	153	1.08	19
p03-f25-v2t2	44	40	1.03	5	36	1.14	14
p03-m25-v1t1	205	273	1.34	5	205	1.15	15
p03-m25-v1t2	2655	x	x	x	x	0	x
p03-m25-v2t1	53	49	1.01	5	43	1.14	10
p03-m25-v2t2	9	9	1.07	0	9	1.02	0
p03-l25-v1t1	15	15	0.99	0	15	1.1	1
p03-l25-v1t2	38	38	0.93	0	36	0.99	1
p03-l25-v2t1	40	40	0.97	0	41	0.98	1
p03-l25-v2t2	37	37	1.04	0	37	1.06	4
p07-f25-v1t1	69	69	0.98	0	44	0.95	1
p07-f25-v1t2	67	67	0.98	0	67	1.01	0
p07-f25-v2t1	158	158	1.02	0	158	0.97	0
p07-f25-v2t2	143	143	1.06	0	143	1.02	0
p07-s25-v1t1	263	253	0.98	1	193	1.03	5
p07-s25-v1t2	81	81	0.97	0	81	0.99	0
p07-s25-v2t1	20	20	1.05	0	20	1.01	1
p07-s25-v2t2	85	83	0.97	1	81	1	5
p07-t25-v1t1	43	43	1.02	0	41	1.08	1
p07-t25-v1t2	463	313	0.85	1	x	x	x
p07-t25-v2t1	165	165	1.03	0	149	1.1	1
p07-t25-v2t2	91	81	0.94	2	69	0.9	7
Average			1.01			1	

Table A.23: Cover and GAP Inequalities for 25-customer Instances: Cont.

Data file	Base Alg.	Alg. with CUTGEN$_{K+G}$			
	Nds	Nds	Time W/WO	# gap cuts	# cover cuts
p01-f25-v1t1	28	24	1.13	1	5
p01-f25-v1t2	81	99	1.25	1	2
p01-f25-v2t1	25	33	1.12	0	1
p01-f25-v2t2	29	41	0.9	0	3
p01-l25-v1t1	29	29	0.97	0	0
p01-l25-v1t2	69	71	1.06	0	1
p01-l25-v2t1	127	132	1.02	0	5
p01-l25-v2t2	33	33	0.99	1	1
p03-f25-v1t1	241	194	0.9	1	34
p03-f25-v1t2	60	56	1.3	1	19
p03-f25-v2t1	151	195	1.13	1	19
p03-f25-v2t2	44	36	1.14	0	14
p03-m25-v1t1	205	197	1.14	2	13
p03-m25-v1t2	2655	x	x	x	x
p03-m25-v2t1	53	43	0.97	2	9
p03-m25-v2t2	9	9	1.03	0	0
p03-l25-v1t1	15	15	1.12	0	1
p03-l25-v1t2	38	36	0.98	0	1
p03-l25-v2t1	40	41	0.98	0	1
p03-l25-v2t2	37	37	1.05	0	4
p07-f25-v1t1	69	44	0.97	0	1
p07-f25-v1t2	67	67	1.05	0	0
p07-f25-v2t1	158	158	0.99	0	0
p07-f25-v2t2	143	143	1.01	0	0
p07-s25-v1t1	263	207	1.03	1	5
p07-s25-v1t2	81	81	1.01	0	0
p07-s25-v2t1	20	20	1.08	0	1
p07-s25-v2t2	85	81	0.99	0	5
p07-t25-v1t1	43	41	1.12	0	1
p07-t25-v1t2	463	x	x	x	x
p07-t25-v2t1	165	149	1.16	0	1
p07-t25-v2t2	91	69	0.9	0	7
Average			1.05		

Vita

Name:	Zeliha Akça.
Place and year of birth:	Cankiri, Turkey, 1979.

Education

- Ph.D. in Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, December 2009.
- M.S. in Management Science, Lehigh University, Bethlehem, May 2005.
- B.S. in Industrial Engineering, Middle East Technical University, Ankara, Turkey, June 2002.

Professional Experience

- Research Assistant (Project Development), Lehigh University, Industrial and Systems Eng. Dept. August 2008 August 2009.
- Research Assistant, Lehigh University, Industrial and Systems Eng. Dept. May 2003 August 2008.
- Instructor, Software Tools Course, Lehigh University, Industrial and Systems Eng. Dept. Spring 2003, Fall2005, Spring 2006, Spring 2007, Fall 2007.

- Teaching Assistant, Lehigh University, Industrial and Systems Eng. Dept. August 2002 - May 2003.
- Summer Intern, Eczacibasi Securities, Turkey, Summer 2001.
- Summer Intern, Vestel Electronics, Manisa, Turkey, Summer 2000.

Publications

- L. Snyder, Z. Akca, Pool Points for Peeps: A network Optimization Model for a Candy Supply Chain, MSOM 2009 Supply Chain Management SIG Conference.
- Z. Akca, R. T. Berger, T. K. Ralphs, A Branch-and-Price Algorithm for Combined Location and Routing Problems Under Capacity Restrictions, in the Proceedings of the Eleventh INFORMS Computing Society Meeting, 2008.
- Z. Akca, R. T. Berger, T. K. Ralphs, Modeling and Solving Location Routing and Scheduling Problems, COR@L Laboratory Technical Report, 2008 (submitted for publication).

Honors and Certification

- Awarded with 1^{st} place in Student Poster Competition in Center For Value Chain Research, Spring Symposium, May 2009.
- Certificates of Excellence & Completion for the Teacher Development Program, June 2008.
- Awarded with Graduate Student of the Year by Lehigh University, Industrial Eng. Dept., May 2006.
- Dean's High Honor List in Middle East Technical University, Turkey, every semester from Fall 1998 to Spring 2002.
- Ranked 63^{rd} in Science and 57^{th} in Mathematics among about 1.5 million high school students in Nationwide University Entrance Exam in Turkey, August 1997.